nature communications

Article

https://doi.org/10.1038/s41467-024-54451-3

Accelerated optimization in deep learning with a proportional-integral-derivative controller

Received: 26 July 2024

Accepted: 8 November 2024

Published online: 26 November 2024

Check for updates

Song Chen[1], Jiaxu Liu[1], Pengkai Wang[2], Chao Xu[2,3,4] ✉, Shengze Cai[2,3] ✉ & Jian Chu[2,3]

High-performance optimization algorithms are essential in deep learning. However, understanding the behavior of optimization (i.e., learning process) remains challenging due to the instability and weak interpretability of the algorithms. Since gradient-based optimizations can be interpreted as continuous-time dynamical systems, applying feedback control to the dynamical systems that model the optimizers may provide another perspective for exploring more robust, accurate and explainable optimization algorithms. In this study, we present a framework for optimization called controlled heavy-ball optimizer. By employing the proportional-integral-derivative (PID) controller in the optimizer, we develop a deterministic continuous-time optimizer called Proportional-Integral-Derivative Accelerated Optimizer (PIDAO), and provide theoretical convergence analysis of PIDAO in unconstrained (non-) convex optimizations. As a byproduct, we derive PIDAO-family schemes for training deep neural networks by using specific discretization methods. Compared to classical optimizers, PIDAO can be empirically proven a more aggressive capacity to explore the loss landscape with lower computational costs due to the property of PID controller. Experimental evaluations demonstrate that PIDAO can accelerate the convergence and enhance the accuracy of deep learning, achieving state-of-the-art performance compared with advanced algorithms.

Deep learning has gained popularity due to the ability of deep function approximation in achieving state-of-the-art performance in a large variety of engineering and scientific tasks. A representative example is learning-based control, where machine learning or deep learning is leveraged to serve complex controlled intelligent systems with remarkable results surpassing those of traditional methods (see Fig. 1a), e.g., motion planning in robotics[1–3], racing[4,5], prediction and control of complex physical systems[6–9], such as fusion plasmas[10,11]. All these deep learning-based tasks primarily rely on solving large-scale optimization problems[12,13], i.e., training deep neural networks (NNs), some of which have over millions or billions of decision variables[14,15]. In general, the train loss landscapes of deep learning models are highly degraded[16], featuring multiple local and global minima that lead to varying generalization capability and model performance[17,18]. In order to obtain suitable minima in the high-dimensional variable space, gradient-based optimization algorithms[19] are generally and successfully employed (Fig. 1a). For example, the gradient descent (GD) and adaptive moment estimation (Adam[20]) algorithms are the most

[1]School of Mathematical Science, Zhejiang University, Hangzhou, Zhejiang, China. [2]Institute of Cyber-Systems and Control, College of Control Science and Engineering, Zhejiang University, Hangzhou, Zhejiang, China. [3]State Key Laboratory for Industrial Control Technology, Zhejiang University, Hangzhou, Zhejiang, China. [4]Zhejiang Provincial Engineering Research Center for Intelligent Mobile Unmanned Systems Technology and Huzhou Key Lab for Autonomous Systems, Huzhou Institute of Zhejiang University, Huzhou, Zhejiang, China. ✉e-mail: cxu@zju.edu.cn; shengze_cai@zju.edu.cn

https://doi.org/10.1038/s41467-024-54451-3

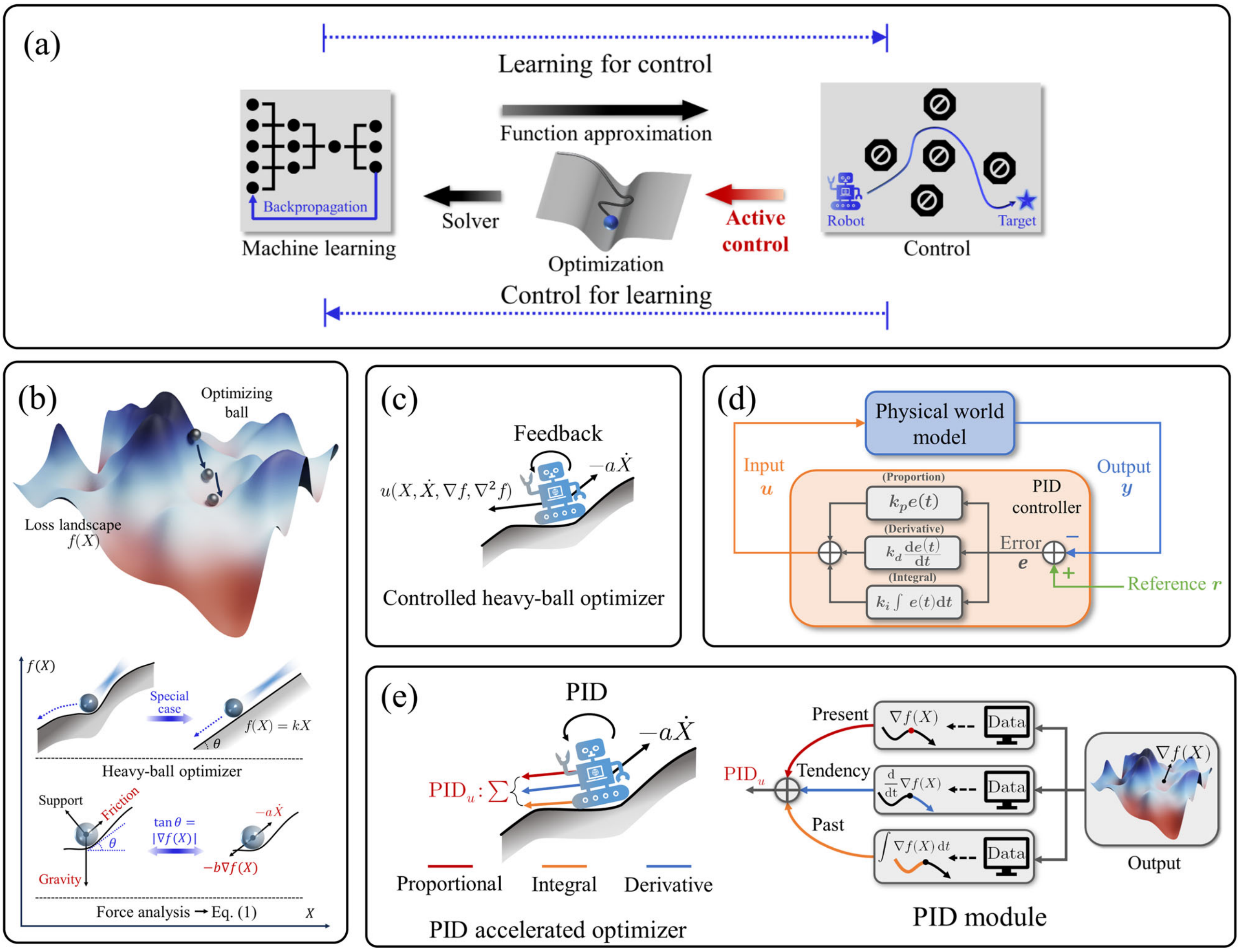

Fig. 1 | Overview of the closed-loop active control optimizer in solving optimization problems. a Setup of the connection between machine learning, control, and optimization. In this work, a perspective of active control is introduced into optimization to further provide a route for control-oriented learning. **b** A physical interpretation for optimization in a physical perspective. This algorithm can be modeled as a passively forced ball using Newton's second law. **c** An illustration of the controlled heavy-ball optimizer. **d** A diagram of the classical PID controller. **e** Architecture of the proposed optimizer, PIDAO, which consists of three key components from the PID mechanism: (1) a proportional part of the "present" output $y(t) = \nabla f(X)$; (2) a derivative of output extracting the "tendency" step; and (3) an integral of output representing an accumulation of the "past".

commonly used optimizers in the deep learning community. To accelerate the training process, physics-inspired mechanisms such as momentum[21] can also be incorporated into the optimization algorithms. There is no doubt that these optimizers contribute mostly to the success of deep learning. However, understanding the impact of optimization on generalization and approximation of deep learning models remains challenging.

A possible solution to understanding the optimization (i.e., learning process) is to model the gradient-based algorithms or NNs as continuous-time dynamical systems[22,23], or discrete-time feedback systems[24–27]. Here, we use the momentum optimizer as an example. Let us consider a differential objective function $f : \Omega \to \mathbb{R}, x \mapsto f(x)$ from a feasible set Ω to the real numbers, where $x \in \Omega$ is the decision variable. The momentum optimizer satisfies the iterative scheme $x_{k+1} = x_k + \beta(x_k - x_{k-1}) - bs\nabla f(x_k)$ with iteration index k, momentum coefficients $\beta > 0$, a step size of s, and a step-scaling factor of $b > 0$. Upon that, for $\min_{x\in\Omega} f(x)$, we can obtain the sequence $\{x_k\}_{k\geq 0}$ generated by the momentum method and compute $f(x_k)$ to approximate the minimum as $k \to +\infty$. By rewriting the momentum method as $\frac{x_{k+1}-2x_k+x_{k-1}}{(\sqrt{s})^2} = -\frac{1-\beta}{\sqrt{s}}\frac{x_k-x_{k-1}}{\sqrt{s}} - b\nabla f(x_k)$, with $\frac{x_{k+1}-2x_k+x_{k-1}}{(\sqrt{s})^2} \approx \ddot{X}(k\sqrt{s})$ and $\frac{x_k-x_{k-1}}{\sqrt{s}} \approx \dot{X}(k\sqrt{s})$, this iterative scheme can be explicitly regarded as an explicit Euler-discretization scheme for the underlying dynamical system:

$$\ddot{X} + a\dot{X} + b\nabla f(X) = 0, \tag{1}$$

where $a \triangleq \frac{1-\beta}{\sqrt{s}}$, $X \triangleq X(t)$ is the state at time t in the continuous-time domain, $\dot{X}$ is the derivative of X with respect to t, and $\nabla f(X)$ is the gradient of f with respect to X. $X(k\sqrt{s})$ approximates x_k as $s \to 0$. Here, the state $X(t)$ and a are tight coupling. Specifically, since Eq. (1) is a second-order system, by using Newton's second law and serving $\ddot{X}$ as an acceleration[21], Eq. (1) can be modeled as a ball with a unit mass rolling across the loss landscape $f(X)$, seeking minima passively driven by a "friction force" $-a\dot{X}$ and a "gravity" $-b\nabla f(X)$ (see Fig. 1b).

Interpreting the gradient-based optimizations as continuous-time dynamical systems allows us to gain insights into the optimization process. By doing this, the phenomena that are not easily explained can be revealed from a physical perspective by building the connection between the properties of learning frameworks (e.g., universal approximation and controllability) and the concepts of control (convergence and Lyapunov stability)[23,27–32]. For example, in addition to the aforementioned momentum optimizer, the commonly-used GD and Adam algorithms have been regarded as continuous-time differential

https://doi.org/10.1038/s41467-024-54451-3

equations[33–35], allowing to reveal the properties of the training dynamics (e.g., generalization and convergence) via Lyapunov stability[29,36–38] and understand the NNs trained using gradient-based algorithms[39,40]. Such a combination of dynamical systems and control theory to understand the optimization and learning process (i.e., control for learning) has attracted increasing attention and has been developing rapidly over the past few years[41–43]. One can introduce open-loop or closed-loop feedback controllers[44,45] into the dynamical systems to mitigate various kinds of deep learning pathologies. For example, an optimal control-based optimizer is proposed that may avoid some pitfalls (e.g., slow convergence on flat loss landscape) of gradient-based methods[24,46,47]. Recently, closed-loop control via feedback has been utilized to enhance the robustness of the training dynamics of generative adversarial networks[48] and to automatically detect and rectify errors caused by perturbations in NNs[49]. The state-feedback law has been also used to mitigate the training unbalance caused by gradient flow[50] and convergence rate problems[51,52]. Following these pioneering works, which provide promising directions on control for learning, we are interested in answering the question in this paper: can the optimization algorithms in deep learning benefit from control theory?

To this end, we integrate active control via feedback into the continuous-time optimization framework (Fig. 1a). First, we recall the momentum optimizer in (Eq. (1)). The ODE (Eq. (1)) can be rewritten as a second-order system $\ddot{X}+a\dot{X}+u=0$ with a control input $u=b\,\nabla f(X)$. Building upon this concept, we propose a framework called controlled heavy-ball optimizer, which is depicted in Fig. 1c and formulated by the following ODE:

$$\ddot{X}+a\dot{X}+u(X,\dot{X},\nabla f(X),\nabla^2 f(X))=0, \tag{2}$$

where $\nabla^2 f(X)$ is the Hessian matrix. From a physical point of view, Eq. (2) is regarded as a mechanical system with active control $u\triangleq u(X,\dot{X},\nabla f(X),\nabla^2 f(X))$ (Fig. 1c). The optimizer with the dynamics (2) seeks optima across the loss landscape $f(X)$ driven by the feedback law u. Unlike the momentum optimizer (Eq. (1)) that is forced to seek optima passively, the optimizer (2) actively employs a feedback control u and subsequently uses feedback to improve its optimization performance, as depicted in Fig. 1c. For the momentum optimizer (Eq. (1)), different values of the coefficient a can impact the convergence rate and the generalization capabilities of the trained models[53], while it is challenging to determine a suitable a due to the tight coupling between a and state $X(t)$. However, by introducing an additional active control u, we can apply various control theories[44] to decouple or minimize this dependency, thus improving the interpretability and stability of the optimizers. Therefore, the Eq. (2) is the basic formation of the optimizer proposed in this paper. The next question is how to design the feedback control u.

Designing the feedback control law is the main issue of control theory[44]. One of the most renowned and efficient feedback controllers in the control society is the proportional-integral-derivative (PID) controller[54]. The PID input describes a sum of the proportional, derivative, and integral actions of the error, where the error is a distance between the output of one physical model and the desired reference, as depicted in Fig. 1d. In fact, the PID controller can reduce the error and enhance the robustness of the systems through the feedback input including the "present" (proportional action), the "past" (integral action), and the "tendency" (derivative action)[55,56]. The idea of PID controller has been used in the design of model architectures[57,58] and optimizer[59]. In particular, for the optimizer, through $\nabla f(X)$, the GD and momentum (Eq. (1)) can be seen as the P-controller which only considers the "present". Wang et al.[59] first introduced the idea of PID into the optimizer design, and proposed a discrete-time optimizer (called PIDopt) by adding a derivative term in the momentum optimizer. However, PIDopt was proposed through qualitative analysis and therefore lacks a rigorous theoretical framework and interpretability.

In this paper, after formulating the optimization problem as a continuous-time control system (Eq. (2)), we quantitatively introduce the PID controller into the design of optimization algorithms to accelerate the learning process and improve the accuracy. By doing this, a more straightforward interpretation of optimization is available due to the interpretability of the PID parametric space. We first employ the PID controller to generate a feedback control input u for the control heavy-ball optimizer (Fig. 1e). As a result, we propose an accelerated optimizer called PID accelerated optimizer (PIDAO) within the continuous-time framework, formulated by Eq. (3) below. PIDAO is more interpretable due to the greater interpretability of the parametric space of the PID control. We present global and local convergence results of PIDAO in unconstrained (strongly) convex and non-convex settings via Lyapunov analysis, respectively, which demonstrate its ability to explore the loss landscape aggressively via a result $\int_{t_0}^{t}\nabla f(X(s))\mathrm{d}s\to 0$. This exploration capability contributes to escape saddle points faster and may converge to better minima during the deep learning training process, such as flat minima that promote improved generalization[16,17]. We use specific discretization methods to derive PIDAO-family schemes for deep learning problems and further validate the effectiveness of the algorithm. Numerical results show that PIDAO inherits the effectiveness of the PID controller in terms of reducing generalization errors and accelerating training convergence with lower computational costs, and outperforms the state-of-the-art methods in various benchmark cases.

We note that our proposal shares a similar idea with PIDopt[59], but differs from PIDopt in terms of the problem formulation, availability of convergence proofs, discrete iterative formulation, and performance for a number of test cases. A detailed comparison between PIDopt and our work is shown in the experiments and in Supplementary Information (SI) Section I.

Results

We first describe the setup of the unconstrained optimization problem. Consider the optimization problem $\min_{X\in\mathbb{R}^n} f(X)$, where f is a differentiable and bounded function. We denote that $f^\star\triangleq\min_{X\in\mathbb{R}^n} f$ and $X^\star\triangleq\mathrm{argmin}_{X\in\mathbb{R}^n} f(X)$. We use $\|\cdot\|$ to represent the standard Euclidean norm hereafter. Three sets of functions are introduced for $f(X)$: $\mathcal{S}^2_{\mu,L}(\mathbb{R}^n)$, $\mathcal{F}^2_L(\mathbb{R}^n)$, and $\mathcal{P}^2_\mu(\mathbb{R}^n)$. In particular, (1) $f\in\mathcal{F}^2_L$ represents that $f(x)$ is convex and twice-differentiable for any $x\in\mathbb{R}^n$ with a L-Lipschitz continuous gradient. (2) $f\in\mathcal{S}^2_{\mu,L}(\mathbb{R}^n)$ (called strongly convex function) means that $f\in\mathcal{F}^2_L$ and $f(X)-\frac{\mu}{2}\|X-X^\star\|^2$ is also convex ($X^\star$ is the minimizer of f). (3) Lastly, $f\in\mathcal{P}^2_\mu$ satisfies the twice-differentiability and the Polyak-Łojasiewicz (PL) condition $\frac{1}{2}\|\nabla f(X)\|^2\geq\mu(f(X)-f^\star)$ for any $x\in\mathbb{R}^n$. Moreover, $C^k(\mathcal{M})$ denotes a space of k-th continuously differentiable functions from space $\mathcal{M}$ to space $\mathcal{M}$. Here, we define three parametric sets: (1) $\mathcal{S}_{\mathrm{PID-SC}}$ and $\mathcal{S}_{\mathrm{PID-NC}}$ are two sets of hyperparameters (k_p, k_i, k_d, a) in the PIDAO, and (2) $\mathcal{S}_{\mathrm{EPID-C}}$ is a set of hyperparameters (k_p, k_i, k_d, a, c) in the Enhanced PIDAO. These three sets are defined in Methods.

Problem Setup: formulations of PIDAO

Before presenting the basic results of the PIDAO for $\min_{X\in\mathbb{R}^n} f(X)$, we first formulate two types of PIDAOs for unconstrained optimization. Firstly, based on Fig. 1d, one basic control input u can be defined as the sum of the proportional, integral, and derivative terms of the output $y(t)=\nabla f(X)$:

$$u=k_p y(t)+k_i\int_{t_0}^{t} y(s)\mathrm{d}s+k_d\frac{\mathrm{d}y(t)}{\mathrm{d}t},$$

where k_p, k_i, and k_d are constants. Then, by combining Eq. (2) and the above input u, for unconstrained optimization, the standard PIDAO is

https://doi.org/10.1038/s41467-024-54451-3

described by the dynamical system:

$$\ddot{X}+a\dot{X}+k_p\nabla f(X)+k_i\int_{t_0}^{t}\nabla f(X)\mathrm{d}s+k_d\frac{\mathrm{d}\nabla f(X)}{\mathrm{d}t}=0, \tag{3}$$

where a, k_p, k_i, and k_d are hyperparamters of this optimizer. Without loss of generality, let us assume that the optimizer starts at $t=t_0$. As discussed in Sec. "Introduction", a control system has various output options, such as $\nabla f(X)$ and $\dot{X}$. Hence, both $\nabla f(X)$ and $\dot{X}$ can be integrated into the output $y(t)$, resulting in $y(t)=c\dot{X}+\nabla f(X)$. Then, by replacing $y(t)=\nabla f(X)$ in (3) with $y(t)=c\dot{X}+\nabla f(X)$, we can obtain another PID-form optimizer called the Enhanced PIDAO. This formulation is mathematically equivalent to

$$\ddot{X}+a\dot{X}+k_p\nabla f(X)+k_i\int_{t_0}^{t}(\nabla f(X)+c\dot{X})\mathrm{d}s+k_d\frac{\mathrm{d}\nabla f(X)}{\mathrm{d}t}=0. \tag{4}$$

In addition, as the integral coefficient k_i in the PIDAO or the Enhanced PIDAO equals 0, this degenerated optimizer is called PDAO, which is formulated by

$$\ddot{X}+a\dot{X}+k_p\nabla f(X)+k_d\frac{\mathrm{d}\nabla f(X)}{\mathrm{d}t}=0. \tag{5}$$

Furthermore, we also consider the possibility of dynamically adjusting the parameters k_p, k_i, k_d in order to better balance or capture the information contained in the output feedback $y(t)=\nabla f(X)$. More discussions about a time-varying PIDAO are presented in SI Section E. For unconstrained optimization $\min_{X\in\mathbb{R}^n}f(X)$, one fundamental question arises regarding Eq. (3)–(5): how to choose hyperparameters k_p, k_i, k_d and a such that the solutions $X(t)$ of Eq. (3)-(5) can satisfy $f(X(t))\to\min_{X\in\mathbb{R}^n}f$ or $\nabla f(X(t))\to 0$ as $t\to+\infty$. We have provided theoretical results for this question in Theorem 1–4 in "Methods". In the following, we demonstrate the main findings of PIDAO for various optimization problems.

Global convergence for convex optimization

Firstly, the strongly convex objective $f\in\mathcal{S}^2_{\mu,L}(\mathbb{R}^n)$ is considered here. The strongly convex objective function is the simplest type of objective function, which contains a global minimum and is often used in regression problems. We present a global convergence result in Theorem 1 in Methods, namely for any $(k_p,k_i,k_d,a)\in\mathcal{S}_{\text{PID}-\text{SC}}$ defined in (6) and $X(t_0)\in\mathbb{R}^n$, there exists a positive constant η such that the solution $X(t)$ of the PIDAO satisfies $\|X(t)-X^\star\|+\|\int_{t_0}^{t}\nabla f(X(s))\mathrm{d}s\|\le O(e^{-\eta t})$. Since $f\in\mathcal{S}^2_{\mu,L}(\mathbb{R}^n)$, it follows that $\|\nabla f(X(t))\|\le O(e^{-\eta t})$ and $f(X(t))-f^\star\le O(e^{-\eta t})$, which eventually means that as the hyperparameters of PIDAO are selected in $\mathcal{S}_{\text{PID}-\text{SC}}$, $f(X(t))$ converges to the global minimum at an exponential rate.

We consider a vanilla quadratic loss as the strongly convex objective $f(X)=0.05X_1^2+5X_2^2$ to verify the performance of the PIDAO, which is demonstrated in Fig. 2a. There exists a line $y=kt+b$ with a finite slope $k<0$ and a finite b such that $\log\left(f(X(t))-f^\star\right)\le kt+b$, which further means that $f(X(t))-f^\star\le O(e^{-kt})$. However, in Fig. 2a, we observe that compared to momentum and PDAO optimizers, PIDAO has a significant overshoot near $X^\star=[0, 0]$, which slows down the convergence for strongly convex optimization. Such a phenomenon is denoted as the offset effect of PIDAO, which will be discussed in Sec. "PIDAO has more potentials to escape local minimum" when explaining the dynamics of PIDAO.

Apart from the basic exponential convergence of PIDAO including $\|X(t)-X^\star\|\le O(e^{-\eta t})$, we remark that the PIDAO achieves an interesting result, namely $\|\int_{t_0}^{t}\nabla f(X(s))\mathrm{d}s\|$ obtains an exponential rate of convergence, which will be discussed in more details in Sec. "PIDAO has more potentials to escape local minimum" as well. To the best of our knowledge, none of the existing gradient-based dynamics, such as the low-resolution dynamics[38,60] and the high-resolution dynamics[29,61], can produce this result, which also cannot be deduced from the inequality $\|\nabla f(X(t))\|\le O(e^{-\eta t})$. To further argue that other optimizers do not have this result, we show a more detailed description in SI Section B.1.

Next, we consider convex objective functions that are more general than strongly convex functions, which appear in machine learning problems such as support vector machines and principal component analysis. We present convergence results of the PIDAO (3) and the Enhanced PIDAO (4) for the convex objective $f\in\mathcal{F}^2_L(\mathbb{R}^n)$. Let us first focus on the convex optimization problem for the Enhanced PIDAO (4). We provide a global convergence result in Theorem 2, which says that for any $(k_p,k_i,k_d,a,c)\in\mathcal{S}_{\text{EPID}-\text{C}}$ defined in (7) and $X(t_0)$, the solution $X(t)$ of the Enhanced PIDAO has $\lim_{t\to+\infty}\int_{t_0}^{t}\left(\nabla f(X(s))+c\dot{X}\right)\mathrm{d}s=\lim_{t\to+\infty}f(X(t))-f^\star=0$. Regarding the second result in Theorem 2, since a strongly convex function $f\in\mathcal{S}^2_{\mu,L}(\mathbb{R}^n)\subset\mathcal{F}^2_L(\mathbb{R}^n)\bigcap\mathcal{P}^2_\mu(\mathbb{R}^n)$, one can also obtain the exponential convergence of the Enhanced PIDAO. Specifically, if $f\in\mathcal{S}^2_{\mu,L}(\mathbb{R}^n)$ and $(k_p,k_i,k_d,a,c)\in\mathcal{S}_{\text{EPID}-\text{C}}$ defined in (7), there exists $\eta>0$ that only depends on hyperparameters such that $\|\int_{t_0}^{t}\left(\nabla f(X(s))+c\dot{X}\right)\mathrm{d}s\|+f(X(t))-f^\star\le O(e^{-\eta t})$.

To verify Theorem 2, we test the performance of Enhanced PIDAO using the same quadratic loss function as in Fig. 2a. We choose three values of c for the optimizer and the exponential convergence results are shown in Fig. 2b. It is also shown that the difference in c has an impact on the convergence rate.

We note that the Theorem 2 is only valid for Enhanced PIDAO. In convex optimization setting, $f\in\mathcal{F}^2_L(\mathbb{R}^n)$, the convergence analysis of the PIDAO (3) cannot be provided in this framework, i.e., $\nabla f(X(t))\to 0$ and $\int_{t_0}^{t}(\nabla f(X(s)))\mathrm{d}s\to 0$ as $t\to+\infty$ cannot be proved together. Fortunately, PIDAO can be approximated by Enhanced PIDAO when c approaches zero (see Proposition 1, 2 in SI Section C). According to such an approximation and Theorem 2, one can use the solution of the Enhanced PIDAO to depict the solution of the PIDAO for convex optimization as precise as possible (see details in SI Section D.2). In Fig. 2b, we show that the difference between PIDAO and Enhanced PIDAO with the same $(k_p,k_i,k_d,a)\in\mathcal{S}_{\text{EPID}-\text{C}}$ can be governed by a small parameter c. As c decreases, the evolution of PIDAO closely aligns with that of Enhanced PIDAO.

In SI Section E, we further investigate a time-varying optimizer for convex optimization with the objective function $f\in\mathcal{F}^2_L(\mathbb{R}^n)$, which provides a unified framework for acceleration and yields an exponential convergence rate similar to that of[32] or a faster convergence rate $O(\frac{1}{t^p}),p>2$ compared to ref. 38 by regulating the prescribed function $g(t)$ defined in SI Section E.

Local convergence for nonconvex optimization

Now we consider the objective f to be twice-differentiable and bounded in $\mathbb{R}^n$ ($f\in\mathcal{C}^2(\mathbb{R}^n)$). Typically, such an objective function is common in neural network training. Additionally, we also consider a class of twice-differentiable functions that satisfy the PL condition ($f\in\mathcal{P}^2_\mu(\mathbb{R}^n)$) in this section, which says that for $\mu>0$, the following holds for the function $f\in\mathcal{P}^2_\mu(\mathbb{R}^n)$: $\frac{1}{2}\|\nabla f(X)\|^2\ge\mu(f(X)-f^\star)$. We note that each stationary point $\hat{X}$ of $f\in\mathcal{P}^2_\mu(\mathbb{R}^n)$ is the global minima[62] since $0=\frac{1}{2}\|\nabla f(\hat{X})\|^2\ge\mu(f(\hat{X})-f^\star)\ge 0$ that implies $f(\hat{X})=f^\star$ and $\hat{X}\in\text{argmin}_{\mathbb{R}^n}f$. An example of non-convex functions satisfying the PL condition is considered: $f(X)=\sum_{k=1}^{n}\left(X_k^2+2\sin^2(X_k)\right)$. PL condition

https://doi.org/10.1038/s41467-024-54451-3

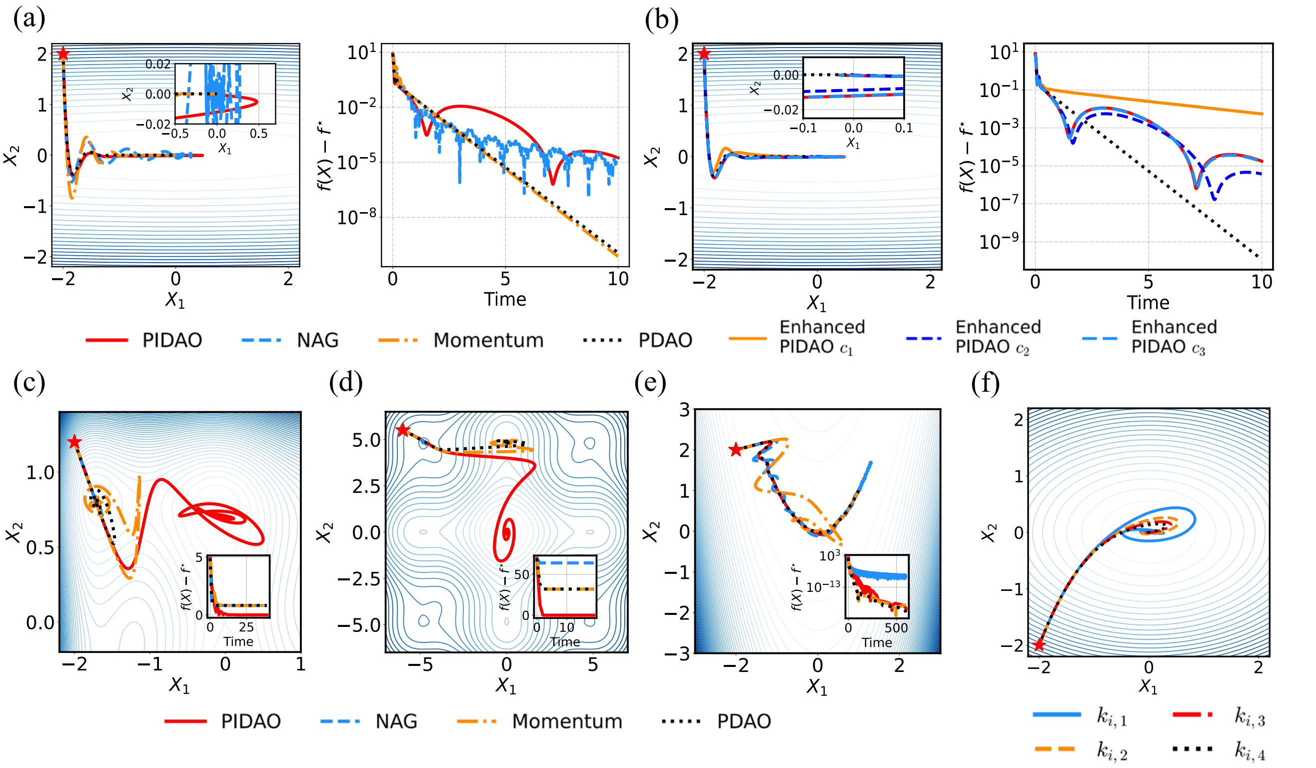

Fig. 2 | The performance of PIDAO for low-dimensional (non)convex optimization. a Evolution trajectories and loss errors of different optimizers with $X(0)=[-2,2]$ for the same ill-condition quadratic loss $f(X)=0.05X_1^2+5X_2^2$. **b** PIDAO can be approximated by the Enhanced PIDAO with a small parameter $c_i (i=1,2,3)$, where they all start from $X(0)=[-2,2]$. Here $c_1=1/2$, $c_2=1/50$, $c_3=1/1000$, respectively. Evolution trajectories of PIDAOs for three test objectives: Six-hump camel back function (**c**), Rastrigin function (**d**), and Rosenbrock function (**e**). **f** Evolution trajectories of four PIDAOs with different k_i for the same quadratic objective function $f(X)=2X_1^2+5X_2^2$ with initial point $X(0)=[-2,-2]$, formulated by (3), where $k_{i,1}>\cdots>k_{i,4}$. The red star in the above figures denotes the initial point $X(0)$. The dynamical systems of optimizers: Nesterov's accelerated gradient optimizer (NAG)[38], Momentum, PIDAO, Enhanced PIDAO, and PDAO, are summarized in Table 3. Furthermore, the hyperparameters of the above cases are given in Methods.

has also received wide attention in the machine learning community[63–65].

We present a local convergence result in Theorem 3 in Methods. For any $(k_p,k_i,k_d,a)\in\mathcal{S}_{\text{PID}-\text{NC}}$ defined in (8) and $X(t_0)\in\mathbb{R}^n$, we have $\lim_{t\to+\infty}\|\nabla f(X(t))\|=0$. We verified the local convergence results under three non-convex losses, as shown in Fig. 2c–e. In all three examples, PIDAO converges to the critical point, but compared to the other optimizers in Fig. 2c, d, PIDAO even converges to the global minimum.

One surprising result in Theorem 3 is that whether the function $f\in\mathcal{C}^2(\mathbb{R}^n)$ is convex or non-convex, the convergence of the critical points ($\nabla f(X(t))\to 0$ as $t\to+\infty$) can be provided. In addition, since we have $\lim_{t\to+\infty}\dot{X}(t)+\frac{k_i}{a}\int_{t_0}^{t}\nabla f(X(s))\mathrm{d}s=0$ in Theorem 3, if $\nabla f(X(t))\to 0$ as $t\to+\infty$ can infer that $X(t)$ converges to a finite point $X^\star$ (namely $\lim_{t\to+\infty}X(t)=X^\star$ and $\|X^\star\|<+\infty$), which implies $\lim_{t\to+\infty}\dot{X}(t)=0$, then one can further get $\lim_{t\to+\infty}\int_{t_0}^{t}\nabla f(X(s))\mathrm{d}s=0$ for nonconvex optimization.

PIDAO has more potentials to escape local minimum

As analyzed above, PIDAO exhibits remarkable behavior in both convex and general settings, either directly or indirectly. It demonstrates the convergence of the integral $\int_{t_0}^{t}\nabla f(X)\mathrm{d}s\to 0$ as t approaches infinity. Based on this deduction, we can suspect that PIDAO possesses an effective capacity to explore the loss landscape $f(X)$ along with the solution trajectory of this optimizer. Qualitatively, for a sufficiently large t meeting with $t-t_0\in\mathbb{N}_+$ and a tagged partition of a closed interval $[t_0,t]$ on the real line given by $t_0\le t_1\le t_2\le\cdots\le t_{n-1}\le t_n=t$ with a unified interval $\delta=t_k-t_{k-1}$, the integral can be approximated by a Riemann sum of a function $f(X(t))$ with respect to this tagged partition. This approximation is defined as:

$$\int_{t_0}^{t}\nabla f(X(s))\mathrm{d}s\approx\sum_{k=0}^{n-1}\nabla f(X(t_k))\delta\approx 0\Rightarrow\underbrace{\sum_{k=0}^{m-1}\nabla f(X(t_k))}_{\text{accumulation}}+\sum_{k=m}^{n-1}\nabla f(X(t_k))\approx 0.$$

Therefore, in the physical sense, the trajectory $X(t)$ of PIDAO needs to cover as much of the loss landscape $f(X)$ as possible in order to use the subsequent accumulation $\sum_{k=m}^{n-1}\nabla f(X(t_k))$ to compensate the previous accumulation $\sum_{k=0}^{m-1}\nabla f(X(t_k))$, eventually making the total accumulation $\sum_{k=0}^{n-1}\nabla f(X(t_k))$ zero. This phenomenon is referred to as the offset effect, which means that PIDAO would not immediately stop searching even in the neighborhood of one local minimum. Instead, it continues to move forward, exploring the loss landscape further. Figure 2f illustrates such offset effect with different values of k_i, where PIDAO equips a more aggressive capacity to explore the loss landscape as k_i increases. Hence, if k_i is appropriately selected, PIDAO may escape local minima or saddle points and converge to a better minimum (even the global minimum).

This property can be further verified by another two examples (Fig. 2c, d), where the objective functions are Six-hump camel back function and Rastrigin function, respectively. Compared with other optimizers, the PIDAO demonstrates the superiority, which can escape from local optimality and converge to global optimality. As shown in the subfigures of Fig. 2c, d, the loss error decreases continuously with respect to time. Regarding the Rosenbrock function (see Fig. 2e), PIDAO goes ahead instead of stopping due to this offset effect, even if it approaches the unique minimum value (see the subfigure at the bottom), which reduces the PIDAO's convergence rate for a short time.

https://doi.org/10.1038/s41467-024-54451-3

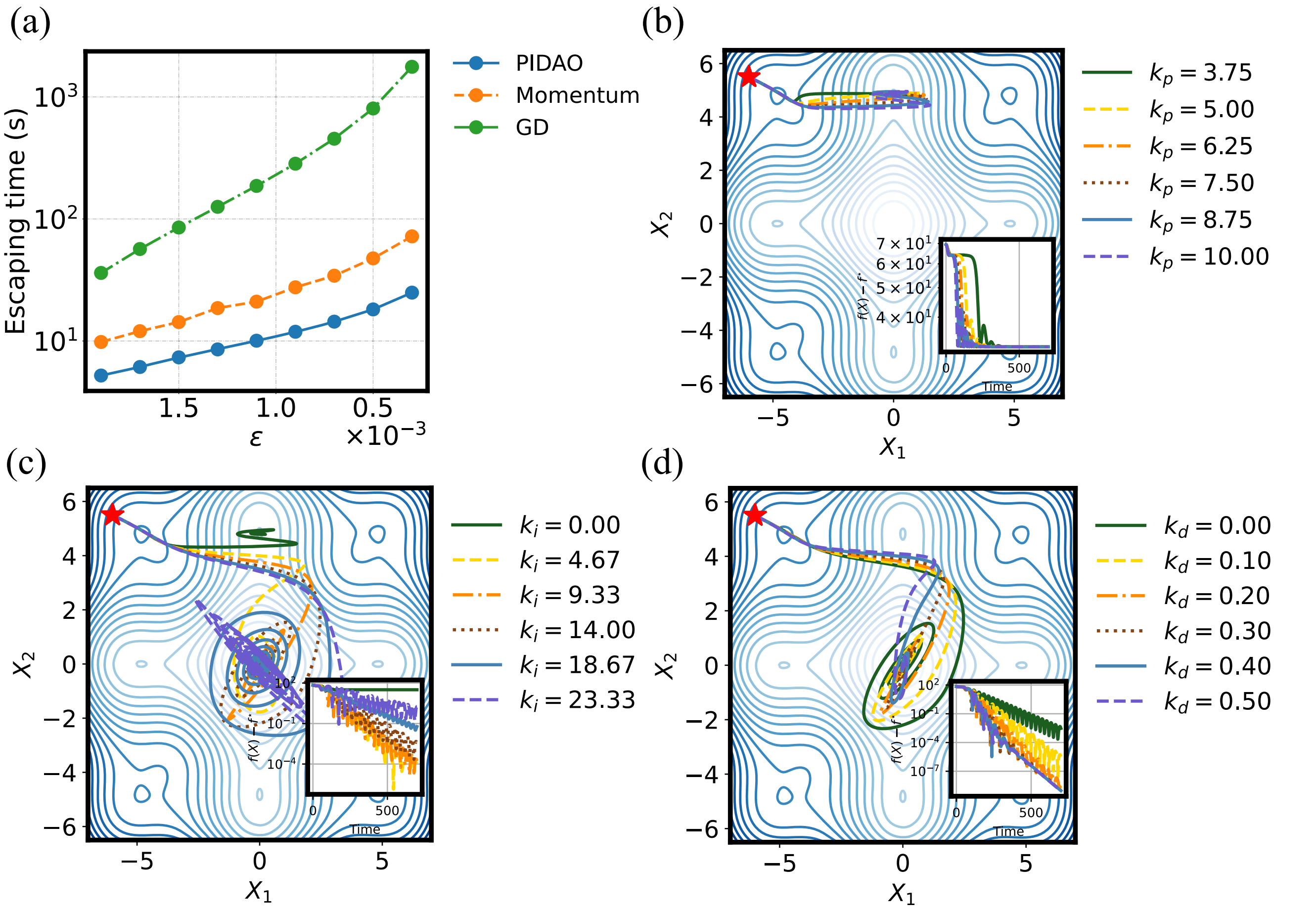

Fig. 3 | PIDAO's ability to escape saddle points and the interpretability of PIDAO's hyperparameters. a Escaping time of different optimizers with the same $X(0) = [0.5, 1]$ for a quadratic function $f(X_1, X_2) = \frac{1}{2}(X_1^2 - \epsilon X_2^2)$. **b**–**d** Evolution trajectories and loss errors of the PIDAO with different hyperparameters (k_p, k_i, k_d) and $X(0) = [-6, 5.5]$ for the Rastrigin loss. **b** Select $k_i = k_d = 0$ but increasing k_p. **c** Select $k_p = 10$ here and $k_d = 0$ but increasing k_i. **d** Select $k_p = 10$ and $k_i = 14$ but increasing k_d. The red star in the above figures denotes the initial point $X(0)$.

This example indicates that if the objective $f(X)$ has a global minimum, the offset effect caused by k_i or the integral may lead to unnecessary oscillation and decrease the convergence rate. Therefore, PDAO, namely the PIDAO without the integral term ($k_i = 0$ in Eq. (3)), would be more suitable for $f(X)$ that has only one minimum (e.g, Rosenbrock function).

The aforementioned results can be concluded by Theorem 4 in Methods. For the strongly convex function $f(X) \in \mathcal{S}^2_{\mu,L}(\mathbb{R}^n)$, PDAO may converge to the minimum faster because there is no additional offset effect arose by k_i. We show for any $(k_p, k_d, a) \in \mathcal{S}_{\text{PD}-\text{SC}}$ and $X(t_0)$, we have a global convergence result of the solution $X(t)$ of PDAO: $f(X(t)) - f^\star \le O(e^{-2mt})$, where $m = \min\{a, \mu k_d\}$. The high-resolution NAG dynamics proposed by Shi et al.[29] is one special case of the PDAO optimizer if $k_p = 1 + \sqrt{\mu s}, k_d = \sqrt{s}, a = 2\sqrt{\mu}$ with $s > 0$. However, our result provides a faster convergence rate $f(X) - \min_{X\in\mathbb{R}^n} f \le O(e^{-\sqrt{2\mu}t})$ rather than $O(e^{-\sqrt{\mu}t/4})$ via $s = 1/2\sqrt{\mu}$. In Fig. 2a, we show that the solution of PDAO converges to the minimum more rapidly than that of the PIDAO, which can be attributed to the absence of the offset effect. Therefore, if the objective function is convex, PDAO is superior to PIDAO which may exhibit critical damping during convergence.

Furthermore, it is shown that PIDAO can escape saddle points at a faster rate than the classical optimizers. In particular, we use a saddle point problem here to discuss PIDAO's ability to escape saddle points. Consider a quadratic objective $f(X_1, X_2) = \frac{1}{2}(X_1^2 - \epsilon X_2^2)$ with a positive constant $\epsilon \ll 1$. We define the escaping time $\tau = \inf\{\tau > 0 \,|\, f(X_1(\tau), X_2(\tau)) < -\delta\}$ for optimizers, where $\delta > 0$ and $X(\tau)$ is the solution of the optimizer. For different optimizers that have the same $X(t_0)$, a smaller escaping time away from the saddle point (0, 0) means a faster-escaping rate. We show that the escaping times τ for PIDAO, momentum, and gradient flow are $O(\frac{1}{\epsilon^{\frac{1}{3}}}\ln\frac{\delta}{\epsilon})$, $O(\frac{1}{\epsilon^{\frac{1}{2}}}\ln\frac{\delta}{\epsilon})$ and $O(\frac{1}{\epsilon}\ln\frac{\delta}{\epsilon})$, respectively, which means that PIDAO can help to escape rather flat saddles and accelerate the optimization process. As shown in Fig. 3a, as ϵ gets smaller (flatter around the saddle point), the escaping times of three optimizers increase, but the escaping time of PIDAO is the smallest. In other words, PIDAO can escape the saddle point with more possibility and less time. More details are provided in SI Section B.2.

Connection between PIDAO and PID controller

We note that there is a clear connection between the PID controller and the PIDAO, which is expected to provide interpretability for PIDAO. In a closed-loop system controlled by a PID controller, the proportional item provides a fast response that allows the state to approach the target quickly. However, there may be a steady-state error, which refers to the difference between the desired setpoint and the actual state of the system. The integral term is utilized to reduce the steady-state error. The derivative item reduces overshoot and oscillation of the state[54,59].

In the context of optimization, we can draw a parallel by considering the steady-state error as the distance between the state $X(t)$ and the global minimum $X^\star$. (1) k_p provides a fast convergence rate for $e(t) = \nabla f(X)$ to converge to zero, which may lead to a steady-state error since the point $X^\star$ with $\nabla f(X^\star) = 0$ may be a local minimum. (2) k_i

https://doi.org/10.1038/s41467-024-54451-3

Table 1 | Algorithms of PIDAO via different discretization methods

Algorithm's name	Algorithm's Scheme (input x_0, $y_0 = k_d \nabla f(x_0)$, and $z_0 = 0$)
PIDAO (SI)	$\begin{cases} x_{k+1} - x_k = hy_{k+1} - hk_d \nabla f(x_k) \\ y_{k+1} - y_k = -hay_{k+1} - h(k_p - ak_d)\nabla f(x_k) - hk_i z_{k+1} \\ z_{k+1} - z_k = h\nabla f(x_k) \end{cases}$
PIDAO (ST)	$\begin{cases} x_{k+1} - x_k = hy_k - hk_d \nabla f(x_k), \\ y_{k+1} - y_k = -hay_{k+1} - h(k_p - ak_d)\nabla f(x_{k+1}) - hk_i z_{k+1}, \\ z_{k+1} - z_k = \frac{h}{2}(\nabla f(x_k) + \nabla f(x_{k+1})), \end{cases}$
PIDAO (AdSI)	$\begin{cases} x_{k+1} - x_k = hy_{k+1} - \frac{hk_d}{\sqrt{\hat{v}_k} + \epsilon} \odot \nabla f(x_k), \\ y_{k+1} - y_k = -hay_{k+1} - \frac{h(k_p - ak_d)}{\sqrt{\hat{v}_k} + \epsilon} \odot \nabla f(x_k) - \frac{hk_i}{\sqrt{\hat{v}_k} + \epsilon} \odot z_{k+1}, \\ z_{k+1} - z_k = h\nabla f(x_k), \\ v_{k+1} - v_k = (1-\rho_1)(\nabla f(x_{k+1}) \odot \nabla f(x_{k+1}) - v_k), \hat{v}_k = \frac{v_k}{1-\rho_1^2}. \end{cases}$

Here, $\odot$ represents the Hadmard inner product and h is learning rate.

reduces the steady-state error. By employing a suitable integral gain k_i in the PIDAO, the exploration capacity of the optimizer can effectively reduce this steady-state error by escaping local minima or saddle points, as discussed earlier. (3) k_d reduces overshoot and oscillation of the optimization process.

Figure 3b–d shows numerical verification of the above interpretability for (k_p, k_i, k_d) in the PIDAO. The Rastrigin loss has local minima $\Omega = \{X|X_1 = -5\sin(X_1), X_2 = -5\sin(X_2)\}$ and $\min_{X\in\mathbb{R}^2} f(X_1, X_2) = f(0,0) = 0$. In Fig. 3b, as k_p increases, the convergence speed of PIDAO to $\Omega = \{X| \nabla f(X) = 0\}$ becomes faster. However, PIDAO finally converges to a local minimum. To overcome this issue, by increasing k_i, PIDAO converges to a global minimum, reducing the steady-state error (Fig. 3c). Moreover, as shown in Fig. 3d, the overshoot and oscillation of the optimization process are eliminated by introducing an appropriate parameter k_d.

PIDAO improves accuracy and efficiency in deep learning

In the aforementioned low-dimensional cases, PIDAO is implemented by directly solving the ODE Eq. (3) with respect to the optimization problems. However, for high-dimensional optimization $\min_{X\in\mathbb{R}^n} f(X)$, it is unrealistic to implement the proposed PIDAO method by solving the ODE within a given error tolerance due to the high computational cost.

To this end, we apply the semi-implicit Euler and symplectic integral methods[66] to develop two iterative forms, that can be easily integrated with the existing deep learning codes, namely PIDAO (SI) and PIDAO (ST); the PDAO scheme is contained in PIDAO (SI) with $k_i = 0$. Moreover, an adaptive algorithm, PIDAO (AdSI), is also proposed by using the adaptive learning rate (*lr*) in PIDAO (SI), inspired by the implementation of RMSprop. These three main algorithms and the corresponding discretization methods are reported in Table 1 and Methods. The hyperparameter h in these algorithms is the learning rate (*lr*). We remark that the gradient term in algorithms can be replaced by the stochastic gradient when using the mini-batch setting. We refer to these algorithms as the PIDAO-family algorithms, which provide alternative approaches to implement the PIDAO in a computationally efficient manner for high-dimensional optimization. In this work, we pay more attention to the implementation of PIDAO-family algorithms. Although the optimization performance such as convergence property cannot be proven theoretically, it is illustrated by empirical experiments in the following. Moreover, suggestions for selecting hyperparameters (k_p, k_i, k_d, a) are provided in Methods.

Example 1: low-dimensional optimization setting. We first compare our method for low-dimensional optimization objectives $f(x)$ with algorithms, namely Momentum, PIDopt, Adam, and Adaptive heavy-ball (called AdaHB)[67] algorithms, in a discrete-time domain. In order to maintain the fairness of the algorithm in comparison, we set the same *lr* and momentum coefficient (if any) in each algorithm. We perform this comparison on two benchmark objectives, quadratic and Rosenbrock, and provide a detailed description of the experimental settings in Methods. The results are summarized in Fig. 4a–b. We find that our method provides comparable or even better results than classical algorithms in each case. In the quadratic objective (Fig. 4a), the PDAO, PIDopt, and Momentum algorithms converge to the minimum faster than the PIDAO (SI) and PIDAO (ST) due to the offset effect of the latters. Notably, the PDAO, PIDopt, and Momentum algorithms exhibit almost identical convergence rates. Regarding the comparison of adaptive algorithms, although PIDAO (AdSI) has a similar trajectory to Adam and AdaHB, the PIDAO (AdSI) holds a faster convergence rate. A similar result can be observed in the Rosenbrock case (Fig. 4b), where we find that all PIDAO-family algorithms converge to smaller loss values, indicating that PIDAO can explore the loss landscape more efficiently.

Example 2: deep learning-based classification. Here, we investigate the performance of PIDAO algorithms on high-dimensional optimization problems using deep learning classifiers as examples. We provide quantitative results for three testing datasets, where each dataset employs a different neural network (NN_θ) to approximate the classifier. Specifically, we utilize fully connected NN (FNN), convolutional NN (CNN), and residual NN (ResNet) architectures for classifying the MNIST (Fig. 4c), FashionMNIST (Fig. 4d), and CIFAR-10 (Fig. 4e) datasets, respectively. For each case, we compare the performance of our algorithms with other algorithms. The evaluation metrics include train loss, test loss, train accuracy, and test accuracy. Further details about the experimental setup can be found in the "Methods" section.

Given the enhanced exploration capability of PIDAO-family algorithms in the loss landscape, we expect that our algorithms can converge to a better minimum with a faster convergence rate. As depicted in Fig. 4c–e, we observe that PIDAO (SI) and PIDAO (ST), with fixed *lr*, consistently outperform the other algorithms in terms of accuracy, loss, and convergence speed in each case. Furthermore, we can conclude that the minimum to which PIDAO (SI) and PIDAO (ST) ultimately converge is flatter than that of other algorithms, leading to better model generalization[17]. On the other hand, regarding the comparison of adaptive algorithms, the results indicate that PIDAO (AdSI) achieves faster convergence than Adam and AdaHB without overfitting, which is confirmed by the higher accuracy in testing. In addition to the aforementioned three cases, we also provide an example where PIDAO is applied to train the PointNet for 3D point cloud classification, which is shown in SI Section H.3, indicating the superior performance of PIDAO as well.

As mentioned above, a key differentiating feature of our algorithms compared with other algorithms is that PIDAO-family algorithms (both fixed-*lr* algorithms and adaptive-*lr* algorithms) converge to flatter minima $\theta^\star$ that lead to better model generalization. To clearly demonstrate this finding, we plot the loss landscapes of those trained models, taking the result of MNIST-FNN as an example. We project the training and test loss landscapes of FNN around the parameter $\theta^\star$ into one- and two-dimensional parameter subspaces[16]; see Fig. 5a, b, respectively. More descriptions of the projection are contained in Methods. As shown in Fig. 5a, we find that in the training results of PIDAO (SI) and PIDAO (ST), the difference between the corresponding train loss function and test loss function is smaller than the difference caused by Momentum and PIDopt algorithms. PIDAO algorithms make the training and testing closer, leading to higher test accuracies (both more than 98.5% on average). Compared to other adaptive algorithms, similar conclusions can be found for PIDAO (AdSI). Furthermore, we see that in the result of train loss shown in Fig. 4c, PIDAO (SI) and PIDAO (ST) present a "loss descent-ascent" phenomenon but end up at a minimum $\theta^\star$ with a train loss similar to that of the Momentum algorithm. We assert this

 https://doi.org/10.1038/s41467-024-54451-3

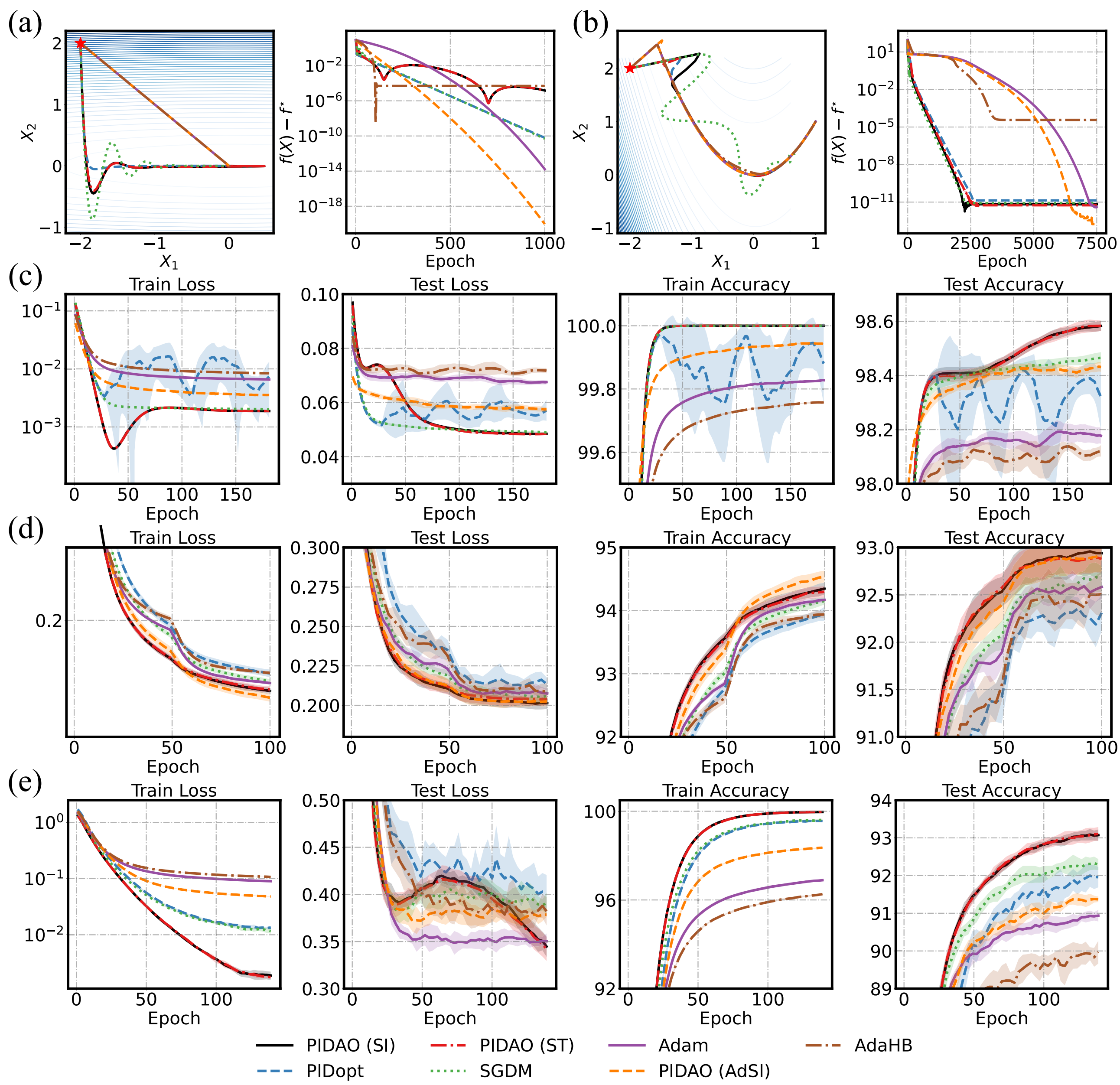

Fig. 4 | Comparison between the classical algorithms and PIDAO-family algorithms on various optimization problems. Convergence trajectories and loss errors of different algorithms on two objective functions, namely the ill-conditioned quadratic loss $f(X) = 0.05X_1^2 + 5X_2^2$ (**a**) and the Rosenbrock function (**b**). The red star in (**a**) and (**b**) denotes the initial point $X(0)$. Performance of different algorithms on classification tasks for MNIST (**c**), Fashion MNIST (**d**), and CIFAR-10 (**e**), respectively. Four evaluation indexes are demonstrated, including train loss, test loss, train accuracy, and test accuracy. The shadows near each curve represent the 90% confidence interval for each data point from 8 experiments.

phenomenon is caused by the effective exploration capacity of PIDAO. Specifically, in Fig. 5b, we project the learning trajectory and train loss landscape of each algorithm into the two-dimensional parameter subspace. During training, both PIDAO (SI) and PIDAO (ST) pass through basins with lower losses (as shown by the red stars in the 30*th* epoch). Instead of terminating the training, they both go ahead and eventually stop at flatter minima (as shown by the blue squares), although the losses are a bit higher than before. This phenomenon is not observed in PIDAO (AdSI) as shown in Fig. 5b, which may be due to the change of continuous-time dynamics of adaptive PIDAO algorithm caused by adaptive parameterization. However, compared with Adam and AdaHB, PIDAO (AdSI) still has better capacity to learn which may be partly inherited from the dynamics of PIDAO.

Example 3: deep learning-based PDE solver. To investigate the capacity of PIDAO-family algorithms on more complex cases such as scientific machine learning, we apply the PIDAO (AdSI) to train NNs for learning partial differential equations (PDEs). The physics-informed NNs (PINNs)[68] and Fourier neural operator (FNO)[69] are investigated. Further details about the experiment are presented in "Methods".

We first consider solving a benchmark fluid dynamics problem via PINNs, namely the steady-state flow in a 2D lid-driven cavity, which is governed by the incompressible Navier-Stokes (NS) equations. The results are summarized in Fig. 6a, where we can observe that the velocity field obtained by the PIDAO (AdSI) is in good agreement with the reference. Moreover, compared to Adam and other optimizers (Fig. 6a), PIDAO (AdSI) presents a higher accuracy and a faster convergence rate. Another example of PINNs is solving the 1D Burgers'

https://doi.org/10.1038/s41467-024-54451-3

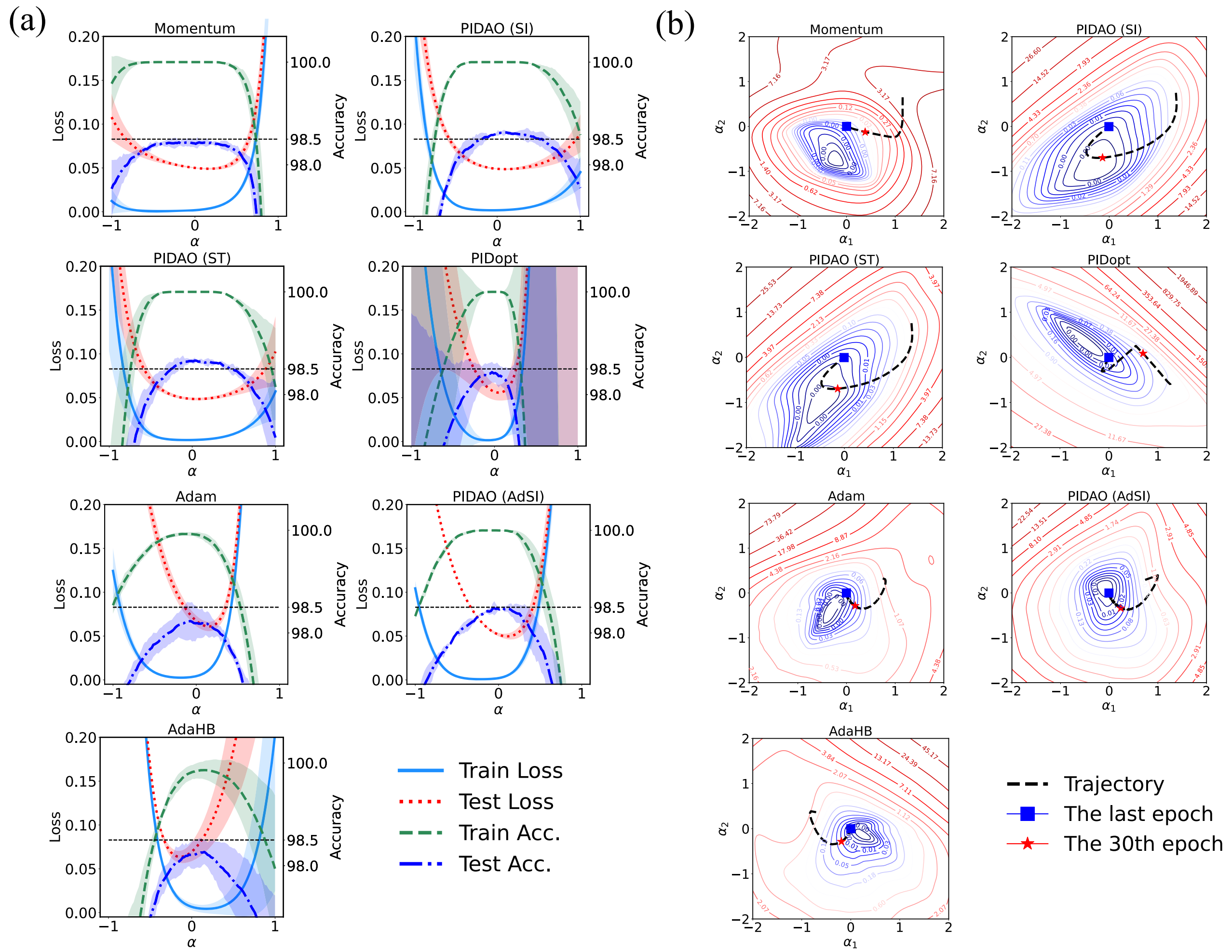

Fig. 5 | One- and two-dimensional subspace projections of loss landscapes in MNIST-FNN case. a Results of one-dimensional projection around the minimum $\theta^{\star}$ with respect to the train loss function, the test loss function, the train accuracy, and the test accuracy. The projection function is formulated as $f(\alpha; \theta^{\star}) = f(\theta^{\star} + \alpha\theta_{pd})$, where f and θ_{pd} are the metric function and the projection direction in the parameter space, respectively. **b** Two-dimensional projection around $\theta^{\star}$ with respect to the train loss function, and the training trajectory. Here, the formula of each projection function is given by $f(\alpha_1, \alpha_2; \theta^{\star}) = f(\theta^{\star} + \alpha_1\theta_{pd_1} + \alpha_2\theta_{pd_2})$ with two projection directions θ_{pd_1} and θ_{pd_2}. The dotted line, red star, and blue square represent the projected training trajectory, the projected point at the 30*th* epoch, and the ended point, respectively. The shadows near each curve represent the 90% confidence interval for each data point from 8 experiments.

equation. As shown in Fig. 6b, we find that the exact and the predicted solutions at different time instants $t = 0.3$, 0.6 are consistent. In addition, the result obtained by PIDAO (AdSI) is more accurate than those of other algorithms, as depicted by the plot of test errors against training epoch in Fig. 6b.

We also investigate the capacity of the proposed algorithm, PIDAO (AdSI), to solve PDEs based on FNO. Here, the 1D Burgers' equation and the 2D Darcy flow equation are considered, the results of which are illustrated in Fig. 6c, d, respectively. From the results, we observe that the models for the two cases trained by PIDAO (AdSI) both converge to lower L_2 losses in testing, which means that PIDAO (AdSI) is more applicable and produces a better performance. The discussion here also provides evidence for the assertion that the PIDAO-family algorithms have more capacity to explore the loss landscape than other algorithms.

All results of the above deep learning examples are summarized in Table 2.

Computational cost of PIDAO

We test the computational cost of PIDAO-family algorithms and the comparison algorithms in two ways: running time per epoch and the running time used to reach a certain train loss (e.g., $1e-3$), which are shown in Fig. 7a, b, respectively. As shown in Fig. 7a, in each task, the PIDAO-family algorithms use a little more runtime per epoch compared to other algorithms, but not by much more. The reason is that PIDAO-family algorithms have to store an integral variable z_k during the iteration, which consumes a certain amount of running time. However, the PIDAO-family of algorithms requires much less runtime than other algorithms to achieve a certain error due to the acceleration of PIDAO. Specifically, in the comparison of adaptive algorithms for the three tasks, PIDAO (AdSI) requires less runtime; in the comparison of non-adaptive algorithms, PIDAO (SI) and PIDAO (ST) also require less runtime, because the PIDAO-family algorithms have a faster convergence rate so that only fewer epochs are needed to reach the specified loss.

Discussion

In this paper, we explore the possibility of applying control theory for deep learning. We establish a connection between closed-loop feedback control and optimization algorithms by the controlled heavy-ball optimizer. This connection not only motivates us to explain the algorithmic behavior and training dynamics through the lens of continuous-time dynamical systems[29,33,38,39,50], but also enables us to develop high-performance closed-loop control optimizers for learning

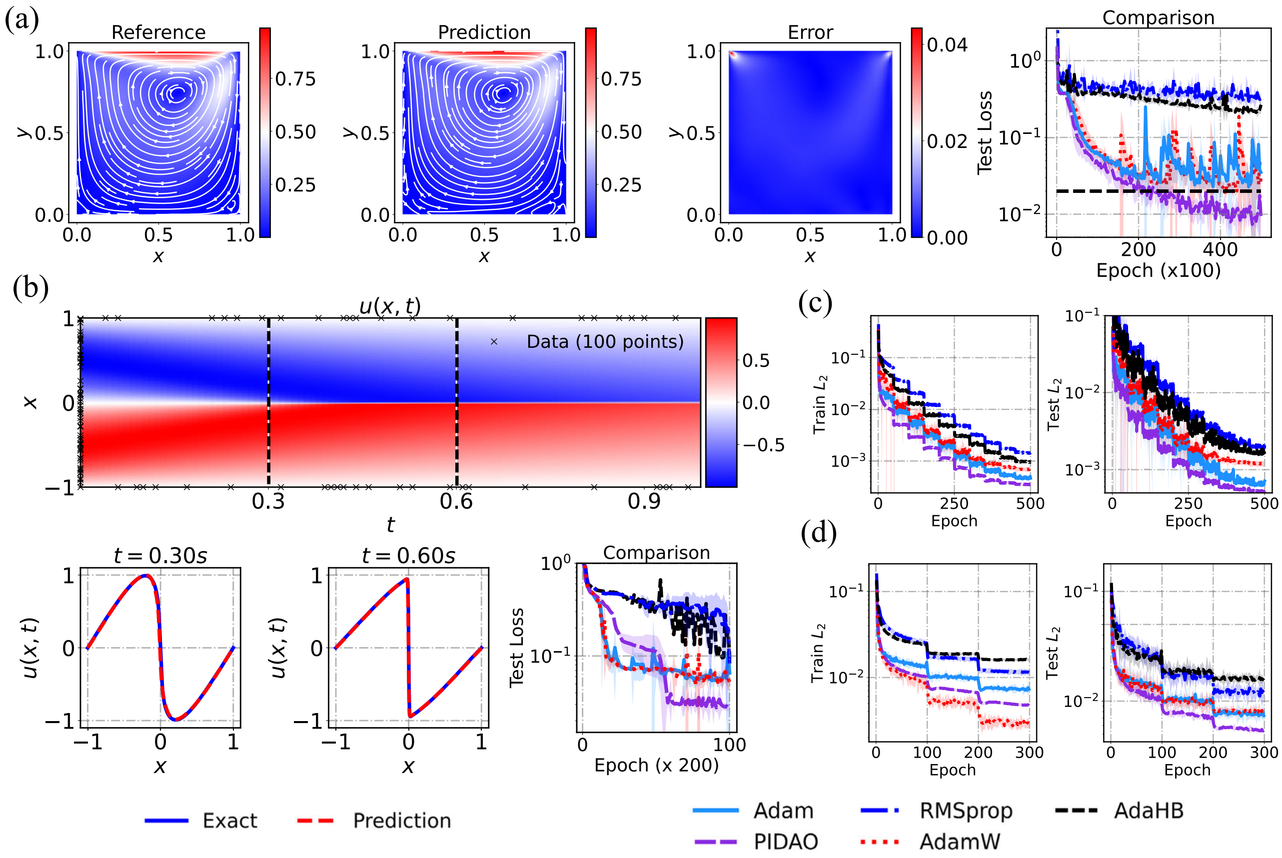

Fig. 6 | Solving partial differential equations via neural networks with different optimizers, including PIDAO (AdSI), Adam, RMSprop, AdamW[75], and AdaHB. **a** Results of 2D lid-driven cavity flow based on physics-informed neural networks (PINNs). The reference flow, predicted flow obtained by PIDAO (AdSI), and the residual flow are plotted. The testing losses against the training epoch based on four optimizers are compared. **b** Results of 1D Burgers' equation solved by PINNs, including the prediction $u(t, x)$ obtained by PIDAO (AdSI) along with the initial and boundary training data, the comparison between the predicted solution and the exact solution at $t = 0.3$, 0.6, and the comparison of test loss between different optimizers. Comparison of the training and testing losses among different optimizers for solving 1D Burgers' equation (**c**) and 2D Darcy equation (**d**) based on Fourier neural operator network. The shadows near each curve represent the 90% confidence interval for each data point from 8 experiments.

Table 2 | Comparison between benchmark algorithms and PIDAO-family algorithms on various deep learning cases

Method	MNIST				Fashion MNIST				CIFAR-10			
	Train Loss	Test Loss	Train Acc	Test Acc	Train Loss	Test Loss	Train Acc	Test Acc	Train Loss	Test Loss	Train Acc	Test Acc
PIDAO (SI)	0.00186	0.04841	100.00%	98.54%	0.15095	0.20203	94.37%	92.89%	0.00205	**0.33433**	**99.97%**	**93.1825%**
PIDAO (ST)	**0.00186**	**0.04831**	**100.00%**	**98.59%**	0.15286	0.20362	94.25%	92.91%	**0.00174**	0.33653	99.97%	93.06%
PIDAO (AdSI)	0.00439	0.05794	99.92%	98.42%	**0.14573**	**0.20131**	**94.58%**	**92.96%**	0.04734	0.37667	98.50%	91.51%
Momentum	0.00201	0.04931	100.00%	98.47%	0.15704	0.20676	94.19%	92.61%	0.01220	0.39213	99.58%	92.25%
Adam	0.00781	0.06894	99.78%	98.20%	0.15615	0.20618	94.20%	92.61%	0.08802	0.34870	96.94%	91.00%
PIDopt	0.00508	0.05739	99.90%	98.29%	0.16225	0.20776	93.99%	92.51%	0.01299	0.40980	99.57%	91.79%
AdaHB	0.00813	0.07174	99.77%	98.09%	0.16418	0.20324	93.93%	92.70%	0.10775	0.37442	96.23%	90.10%

Method	PINNs for 1D Burgers' equation		PINNs for 2D cavity flow			FNO for 1D Burgers' equation		FNO for 2D Darcy equation	
	Train Loss	Test Error of $u(x, t)$	Train Loss	Test Error of $u(x, y)$	Test Error of $v(x, y)$	Train Loss	Test L_2 Error of $u(x, t)$	Train Loss	Test L_2 Error of $u(x_1, x_2)$
PIDAO (AdSI)	5.8440×10^{-4}	**2.87%**	$\mathbf{6.3204 \times 10^{-5}}$	**1.30%**	**1.76%**	$\mathbf{3.4827 \times 10^{-4}}$	$\mathbf{5.3612 \times 10^{-4}}$	1.1918×10^{-2}	$\mathbf{5.4192 \times 10^{-3}}$
RMSprop	9.2526×10^{-3}	12.20%	8.9772×10^{-3}	34.17%	68.80%	1.4479×10^{-3}	2.0741×10^{-3}	1.7722×10^{-2}	1.2003×10^{-2}
Adam	5.5540×10^{-4}	6.58%	1.8461×10^{-4}	3.07%	4.22%	4.6665×10^{-4}	7.1373×10^{-4}	1.0586×10^{-2}	7.7934×10^{-3}
AdamW	$\mathbf{3.4320 \times 10^{-4}}$	6.23%	1.4487×10^{-4}	2.51%	3.89%	7.0761×10^{-4}	1.2057×10^{-3}	$\mathbf{5.8096 \times 10^{-3}}$	8.2432×10^{-3}
AdaHB	3.8605×10^{-3}	9.43%	3.1194×10^{-3}	21.07%	43.73%	9.5934×10^{-4}	1.7213×10^{-3}	1.9964×10^{-2}	1.6065×10^{-2}

The best results are marked in bold. The hyperparameter settings of these optimizers can be found in SI Section H.

https://doi.org/10.1038/s41467-024-54451-3

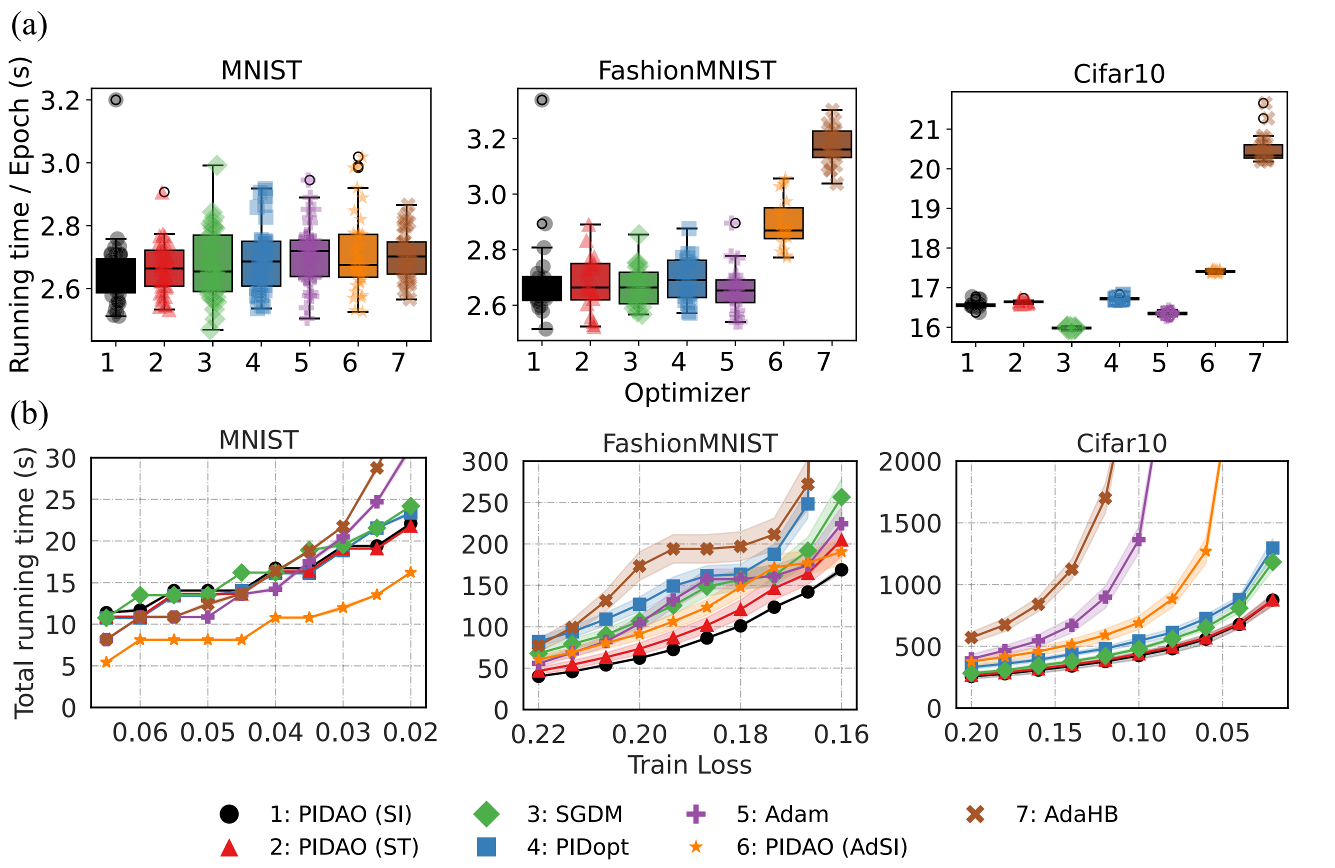

Fig. 7 | Comparison of computational cost between the classical algorithms and PIDAO-family algorithms on data classification problems. a The running time used per epoch of optimizers. **b** The running time of optimizers used to reach a certain threshold of train loss. The shadows near each curve represent the 90% confidence interval for each data point from 8 experiments.

via feedback. In this work, by leveraging the PID control, we propose an accelerated optimizer, PIDAO. We investigate the convergence property and the performance of PIDAO for three basic types of unconstrained optimization problems: strongly convex, convex, and bounded non-convex optimization. Most of the characteristics are described with theoretical analysis. We prove that PIDAO can achieve an exponential convergence in the strongly convex setup and achieve global and local convergence results in the convex and non-convex setups, respectively. Our numerical experiments demonstrate that PIDAO exhibits a more aggressive exploration capacity across the loss landscape. Such a property provides PIDAO the potential to faster escape from suboptimal minima and converge towards superior minima, such as global minima or flat minima. Eventually, to further improve the applicability of PIDAO, we generate the PIDAO-family algorithms via discretization methods, including two learning rate-fixed algorithms and one adaptive algorithm, which can be easily employed in deep learning tasks. As demonstrated in numerous examples (see Figs. 4–7), our algorithms possess a more effective exploration competence along the loss landscape with lower computational costs, resulting in improved accuracy and efficiency in deep learning applications (e.g., classification and PDE-solver).

Our proposed framework is inspired by a second-order ODE (1) characterizing the heavy-ball algorithm. However, there are notable distinctions between PIDAO and other existing optimizers, which are summarized in Table 3. The primary differentiation lies in the generation of each optimizer. Most optimizers are typically derived by establishing an iteration scheme and then deducing the corresponding continuous-time optimizer through limit analysis. However, PIDAO is first introduced by incorporating a controller into the momentum optimizer from a physical perspective, followed by the implementation of several PIDAO-family algorithms through discretization. In addition, there are also differences in the way how the optimizers handle the current, past, and tendency information. The momentum optimizer employs the difference between the current state and the previous state for acceleration, and the adaptive optimizers (e.g., Adam) construct the first and second moments to process the past and current gradient information, and further adaptively update the search direction in the solution space and the learning rate. On the contrary, PIDAO captures current, past, and tendency from the gradient information, corresponding to the proportional, integral, and derivative computations in the PID controller. This distinctive approach makes PIDAO more comprehensive and interpretable in those learning and optimization problems. Moreover, compared with another PID-inspired optimizer, PIDopt[59], the motivation of PIDAO is different. We also propose PIDAO with theoretical results in the continuous-time framework. In addition, PIDAO is a more general framework, and PIDopt can be considered as a typical PD "controller" in PIDAO. We provide more details on the comparison between PIDAO and PIDopt in SI Section I.

It is worth noting again that PIDAO achieves an interesting result $\int_{t_0}^{t} \nabla f(X(s))\mathrm{d}s \to 0$ as $t \to +\infty$, which is called offset effect and is a piece of evidence that the PIDAO has a stronger exploration capability along the loss landscape. We set up numerous experiments, including low-dimensional optimization problems and high-dimensional learning tasks, to verify the effectiveness of the proposed optimizer. In particular, the results of complex deep-learning tasks indicate that our algorithms can outperform state-of-the-art algorithms (e.g., Adam) in terms of accuracy.

The framework of PIDAO is constructed based on the PID mechanism in this study. We believe there is substantial room for

https://doi.org/10.1038/s41467-024-54451-3

Table 3 | Comparison between existing optimizers and PIDAO in the continuous-time perspective

Optimizers	Continuous-time dynamical systems of the optimizers
GD	$\dot{X}+k_p\nabla f(X)=0$
Momentum[21]	$\ddot{X}+a\dot{X}+k_p\nabla f(X)=0$
NAG[38]	$\ddot{X}+\frac{3}{t}\dot{X}+k_p\nabla f(X)=0$
High-resolution NAG-C[29]	$\ddot{X}+\frac{3}{t}\dot{X}+\left(1+\frac{3\sqrt{s}}{2t}\right)\nabla f(X(t))+\sqrt{s}\frac{\mathrm{d}}{\mathrm{d}t}\nabla f(X)=0$
High-resolution NAG-SC[29]	$\ddot{X}+2\sqrt{\mu}\dot{X}+(1+\sqrt{\mu s})\nabla f(X)+\sqrt{s}\frac{\mathrm{d}}{\mathrm{d}t}\nabla f(X)=0$
Adaptive optimizer[35]	$\begin{cases}\dot{X}=-m(t)/\sqrt{v(t)+\varepsilon}\\ \dot{m}=h(t)\nabla f(X)-r(t)m\\ \dot{v}=p(t)[\nabla f(X)]^2-q(t)v\end{cases}$
PDAO	Eq. (3) with k_i = 0
PIDAO	Eq. (3)
Enhanced PIDAO	Eq. (4)

future work in applying other existing but suitable control methods to construct controlled physical optimizers (e.g., heavy ball in Sec. "Introduction"), that can efficiently solve learning problems. Although this article focuses on unconstrained optimization issues, in the future, we could study constrained optimization problems. For example, it is possible to consider physical systems for optimization on manifolds and their control problems, such as the control of Euler-Lagrange systems on Riemannian manifolds. Regarding PIDAO, there are still several aspects that require further investigation. The first one is the optimal convergence rate of PIDAO for strongly convex optimization. Although both the PIDAO family of learning rate-fixed algorithms and adaptive algorithms outperform other algorithms in the examples here, we do not yet know why they improve the accuracy of learning. A possibility would be to explore the relationship between a key feature of the proposed algorithms, namely the offset effect, and the generalization ability of deep learning models, drawing upon well-established aspects[39,70]. Moreover, although the algorithm PIDAO (AdSI) achieves great performance in the deep learning examples, the continuous-time dynamical system of PIDAO (AdSI) is no longer exactly equivalent to PIDAO due to the incorporation of an adaptive learning rate. Therefore, deducing a continuous-time system for PIDAO (AdSI) would be a promising research direction to gain further insights into this adaptive algorithm. Last but not least, the concept of PIDAO can be extended to continuous-time stochastic differential equations, as mini-batch algorithms are commonly employed in deep learning practice.

Methods

Here, we describe more details about the proposed optimizer PIDAO, and summarize the discretization methods of PIDAO, the hyperparameters, testing objective functions, and NN architectures that we used in our numerical experiments. We note that in this paper, the continuous-time optimizers for low-dimensional optimization in Fig. 2 and the discrete-time algorithms for deep learning in Figs. 4–6, are implemented in the `solve_ivp` method and the `Pytorch` package, respectively. Our implementations are all publicly available.

Global convergence for strongly convex optimization

We denote one nonempty set of hyperparameters (k_p, k_i, k_d, a):

$$\mathcal{S}_{\text{PID}-\text{SC}}=\left\{(k_p,k_i,k_d,a)\in\mathbb{R}_+^4\,|k_p>ak_d,\mu k_d^2+ak_d>k_p+\frac{k_i}{a}\right\},\qquad(6)$$

Theorem 1. We suppose that $f\in\mathcal{S}_{\mu,L}^2(\mathbb{R}^n)$. For any $(k_p,k_i,k_d,a)\in\mathcal{S}_{\text{PID}-\text{SC}}$ defined in (6), there exists a positive constant η such that the solutions of the proposed PIDAO (3), $X:[t_0,+\infty)\to\mathbb{R}^n$, satisfy $\|X(t)-X^\star\|+\|\int_{t_0}^t\nabla f(X(s))\mathrm{d}s\|\le O(e^{-\eta t})$, where η only depends on parameters k_p, k_i, k_d, a, μ.

The proof of Theorem 1 is given in SI Section A.1. We also provide a stochastic analysis result of PIDAO in this setting in SI Section A.2. We highlight the following points. (i) One key aspect in the proof of Theorem 1 is the construction of a general Lyapunov function $V(X)$[44], as shown in SI Section A.1. The set $\mathcal{S}_{\text{PID}-\text{SC}}$ ensures that $V(X)$ exponentially decreases along with the solution $X(t)$ of (3), which means that different Lyapunov functions $V(X)$ may require different $\mathcal{S}_{\text{PID}-\text{SC}}$ for convergence. Hence, the set $\mathcal{S}_{\text{PID}-\text{SC}}$ is conservative. (ii) η is a nonlinear function of k_p, k_i, k_d, a, μ. In other words, the convergence rate is determined in a nonlinear manner by the parameters k_p, k_i, k_d, a, μ. The parameters leading to an optimal convergence rate for strongly convex optimization[71] would not be discussed in this work. However, in Methods, we summarize how to select k_p, k_i, k_d, a in practical applications.

Global convergence for convex optimization

We propose another set for hyperparameters (k_p, k_i, k_d, a, c):

$$\mathcal{S}_{\text{EPID}-\text{C}}=\left\{(k_p,k_i,k_d,a)\in\mathbb{R}_+^4\,|k_p>\max\left\{\frac{k_i+\frac{k_d^2L^2}{4}}{a},\frac{k_dk_i}{r}+r,a\right\},\frac{k_i}{r}+rc=a,0<c<1\right\},\qquad(7)$$

where $r>0$ is related with k_i, c, a by the second equality in (7).

Theorem 2. For the solution of the proposed Enhanced PIDAO $X:[t_0,+\infty)\to\mathbb{R}^n$, if $f\in\mathcal{F}_L^2(\mathbb{R}^n)$ and $(k_p,k_i,k_d,a,c)\in\mathcal{S}_{\text{EPID}-\text{C}}$ defined in (7), then the following assertions hold:

(1) For all $X(t_0)\in\mathbb{R}^n$, the global convergence can be held, i.e.,

$$\lim_{t\to+\infty}\int_{t_0}^t\left(\nabla f(X(s))+c\dot{X}\right)\mathrm{d}s=\lim_{t\to+\infty}f(X(t))-\min_{X\in\mathbb{R}^n}f=0.$$

(2) Moreover, if the convex objective f obeys the Polyak-Łojasiewicz condition with $\mu>0$, i.e., $f\in\mathcal{F}_L^2(\mathbb{R}^n)\bigcap\mathcal{P}_\mu^2(\mathbb{R}^n)$, there exists M, $\eta>0$ such that the exponential convergence rate can be held as follows: $\|\int_{t_0}^t(\nabla f(X(s))+c\dot{X})\mathrm{d}s\|+(f(X(t))-\min_{X\in\mathbb{R}^n}f)\le Me^{-\eta t}$, where M and η only depend on parameters k_p, k_i, k_d, a, L, μ.

We provide the proof in SI Section D.1.

Local convergence for non-convex optimization

Now we define a set of $(k_p,k_i,k_d,a)\in\mathbb{R}_+^4$ for this class of objective,

$$\mathcal{S}_{\text{PID}-\text{NC}}=\left\{(k_p,k_i,k_d,a)\in\mathbb{R}_+^4\,|k_p-\frac{k_i}{a}-ak_d>0\right\}.\qquad(8)$$

Theorem 3. We consider the parameters $(k_p,k_i,k_d,a)\in\mathcal{S}_{\text{PID}-\text{NC}}$ defined in (8) and the bounded objective function $f\in\mathcal{C}^2(\mathbb{R}^n)$. Then, the solution trajectory of PIDAO (3) with the respective parameters (k_p, k_i, k_d, a) starting from any initial condition $X(t_0)$ holds

$$\lim_{t\to+\infty}\nabla f(X(t))=\lim_{t\to+\infty}\dot{X}(t)+\frac{k_i}{a}\int_{t_0}^t\nabla f(X(s))\mathrm{d}s=0.$$

Moreover, if $f\in\mathcal{P}_\mu^2(\mathbb{R}^n)$, the global convergence of the global minima can be obtained, i.e., $\lim_{t\to+\infty}f(X(t))-f^\star=0$ for any $X(t_0)\in\mathbb{R}^n$.

More details of the proof can be found in SI Section F.

https://doi.org/10.1038/s41467-024-54451-3

Fast convergence of the PDAO optimizer for strongly convex optimization

Let us define a set for (k_p, k_d, a) in the PDAO:

$$\mathcal{S}_{\mathrm{PD-SC}} = \left\{(k_p, k_d, a) \in \mathbb{R}_+^3 \,|\, k_p > a k_d\right\}. \tag{9}$$

Theorem 4. We assume that $f \in \mathcal{S}_{\mu,L}^2(\mathbb{R}^n)$. For any $(k_p, k_d, a) \in \mathcal{S}_{\mathrm{PD-SC}}$ defined in (9), the solution $X : [t_0, +\infty) \to \mathbb{R}^n$ of the PDAO formulated by Eq. (3) with $k_i = 0$ satisfies $f(X) - f^\star \le O(e^{-2mt})$, where $m = \min\{a, \mu k_d\}$.

We provide the proof in SI Section G.

Discrete-time algorithms of PIDAO

As shown in Proposition 1 in SI Section C, the following first-order ODE group is equivalent to the original ODE (Eq. (3)):

$$\text{Equivalent first-order ODE}: \begin{cases} \dot{X} = Y - k_d \nabla f(X), \\ \dot{Y} = -aY - \left(k_p - a k_d\right) \nabla f(X) - k_i Z, \\ \dot{Z} = \nabla f(X). \end{cases}$$

Then, we can discrete these ODEs to solve the optimization problems. In this paper, three discrete-time PIDAO algorithms are proposed and implemented in `Pytorch`, which are summarized in Table 1.

PIDAO (SI)

The semi-implicit Euler method is used to obtain PIDAO (SI), which means that on the right side of the above ODE, we use explicit x_k, and implicit y_{k+1}, z_{k+1} to drive the scheme. As $k_i = 0$, we also produce the scheme of PDAO.

PIDAO (ST)

The algorithm PIDAO (ST) is based on the symplectic method and the trapezoid formula: in detail, the first and second equations in the above ODE are discretized by the symplectic method, while the third equation uses the trapezoid formula.

PIDAO (AdSI)

Based on the PIDAO (SI) scheme, we exploit an adaptive algorithm PIDAO (AdSI) by introducing the adaptive *lr* factor $\sqrt{\hat{v}_k}$ and the moving average into the gradient direction $\nabla f(x)$ and z_k. Here, instead of using all the past gradients in the updated z_k, we use a moving average of past gradients which is inspired by RMSprop. In particular, we calculate the exponential weighted moving average of gradient squared $v_{k+1} = v_k + (1 - \rho_1)(\nabla f(x_{k+1}) \odot \nabla f(x_{k+1}) - v_k)$ and use a correction of v_k, i.e., $\hat{v}_k = \frac{v_k}{1-\rho_1^2}$, to update k_p, k_i, k_d.

The pseudo-codes for these three algorithms are provided in Algorithm 1–3 in SI Section H.

Hyperparameters in discretized PIDAO

To further improve the practicality of the algorithm, we explore the choice of hyperparameters in the above PIDAO-family algorithms (shown in Table 1). Recall the PIDAO (SI), we can rewrite the corresponding scheme by substituting $y_{k+1} = \frac{x_{k+1} - x_k}{h} + k_d \nabla f(x_k)$ into the second equation in this scheme and generate the following scheme:

$$x_{k+1} - x_k = \frac{1}{1+ah}(x_k - x_{k-1}) - \frac{h^2 k_p}{1+ah} \nabla f(x_k) - \frac{h k_d}{1+ah} (\nabla f(x_k) - \nabla f(x_{k-1})) - \frac{h^2 k_i}{1+ah} z_{k+1}.$$

By setting $k_d = k_i = 0$, the PIDAO (SI) degrades into the momentum scheme $x_{k+1} - x_k = \frac{1}{1+ah}(x_k - x_{k-1}) - \frac{h^2 k_p}{1+ah} \nabla f(x_k)$. Here, we serve two terms $\frac{1}{1+ah}$ and $\frac{h^2 k_p}{1+ah}$ as the equivalent momentum and the equivalent learning rate, denoted by a_{eq} and lr_{eq} respectively, which can produce suitable k_p, a, h in this view. For example, if we take the equivalent momentum and the equivalent learning rate as $a_{eq} = 0.9$ and $lr_{eq} = 0.01$, respectively, then we have that $h = lr_{eq} = 0.01$, $a = \frac{1}{h}(\frac{1}{a_{eq}} - 1) = \frac{100}{9}$, and $k_p h^2 a_{eq} = lr_{eq} \Rightarrow k_p = \frac{1}{a_{eq} lr_{eq}} = \frac{1000}{9}$. For the remaining two hyperparameters in the PIDAO (SI), k_d and k_i, their selection needs to satisfy $k_p > \frac{k_i}{a} + a k_d$ such that $k_p, k_i, k_d, a \in \mathcal{S}_{\mathrm{PID-NC}}$ that is defined in (8).

Therefore, we summarize the method of selecting hyperparameters: once we determine a_{eq} and lr_{eq}, (k_p, k_i, k_d, a) and h could follow that

$$h = lr_{eq},\ a = \frac{1}{lr_{eq}}\left(\frac{1}{a_{eq}} - 1\right),\ k_p = \frac{1}{a_{eq} lr_{eq}},\ k_p > \frac{k_i}{a} + a k_d.$$

The above parameter selection method is also applicable to the PIDAO (ST) and the PIDAO (AdSI). In practice, we suggest that $k_i \in (0, 10)$ and $k_d \in (0, 1]$. Further optimization for hyperparameters k_i and k_d can be based on Bayesian optimization or rich PID tuning methods in classical control theory, such as Ziegler-Nichols tuning[54]. The default a_{eq} and lr_{eq} for PIDAO (SI) and PIDAO (ST) are suggested by $lr_{eq} = 10^{-2}$, $a_{eq} = 0.9$, while for PIDAO (AdSI) $lr_{eq} = 10^{-3}$, $a_{eq} = 0.9$ are available in practice.

Experiments on low-dimensional optimization via continuous-time setting

The continuous-time optimizers investigated in Fig. 2 are summarized in Table 3. The corresponding ODE for each optimizer is solved using the backward differentiation formula (BDF) method. Firstly, we introduce the hyperparameters of PIDAO used in strongly convex cases (Fig. 2a, f). In Fig. 2a and b, we use $k_p = \frac{1000}{9}, k_i = 60, k_d = 2.5, a = \frac{100}{9}$ for PIDAO and Enhanced PIDAO. In Fig. 2f, for one function $f \in \mathcal{S}_{\mu,L}^2(\mathbb{R}^n)$, $f - \mu\|X - X^\star\|^2/2$ is a convex function. By choosing suitable k, s, ϵ, one can construct the following subset of $\mathcal{S}_{\mathrm{PID-SC}}$:

$$\mathcal{S}_{\mathrm{PID-SC}}^{\mathrm{sub}}(k, s, \epsilon) = \left\{k_d = k\sqrt{s}, a = 2\sqrt{\mu}, k_p = 2k\sqrt{\mu s} + \epsilon, 0 < k_i < (\mu k_d^2 + a k_d - k_p)a - \epsilon\right\}.$$

For example, in Fig. 2f, we derive $\mu = 4$ and select $k = 9, s = 0.01, \epsilon = 10^{-4}$ that produces $k_d = 0.9, a = 4, k_p = 3.6001, k_{i,n} = \frac{12.9595}{n}$. Moreover, we use the same k_p, k_d, a (if any) in the PDAO, NAG, and Momentum optimizers due to the fairness.

Secondly, for non-convex cases (Fig. 2c–e), the hyperparameters of PIDAO should obey $\mathcal{S}_{\mathrm{PID-NC}}$(8). Here, we consider three objectives for non-convex optimization, namely (1) Six-hump camel back (SHCB) $f(X) = (4 - 2.1X_1^2 + \frac{X_1^4}{3})X_1^2 + X_1X_2 + (-4 + 4X_2^2)X_2^2$, (2) Rosenbrock (ROS) $f(X) = 20(X_2 - X_1^2)^2 + (X_1 - 1)^2$, and (3) Rastrigin (RAS) $f(X) = 20 + X_1^2 + X_2^2 - 10(\cos(X_1) + \cos(X_2))$. k_p, k_d, and a are the same in these cases: 10, 0.3, and 3, respectively, while k_is are different, which are 28 (for SHCB), 9 (for ROS), and 9 (for RAS), respectively.

Experiments on low-dimensional optimization via discrete-time setting

We compare the proposed PIDAO-family algorithms with other algorithms in two cases, namely the ill-conditioned quadratic loss $f(X) = 0.05X_1^2 + 5X_2^2$ and the Rosenbrock objective $f(X) = 20(X_2 - X_1^2)^2 + (X_1 - 1)^2$. After selecting the equivalent momentum a_{eq} and the equivalent learning rate lr_{eq}, the h and k_p, k_i, k_d, a in PIDAO-family algorithms follow the design method of hyperparameters illustrated

https://doi.org/10.1038/s41467-024-54451-3

above $h = lr_{eq}$, $a = \frac{1}{lr_{eq}}(\frac{1}{a_{eq}} - 1)$, $k_p = \frac{1}{a_{eq} lr_{eq}}$, $k_p > \frac{k_i}{a} + ak_d$. And the learning rate and momentum coefficients α (if any) are also used for Adam and Momentum algorithms via $h = lr_{eq}$, $\alpha = a_{eq}$.

Experiments on deep learning-based classification

We compare the PIDAO-family algorithm with other comparison algorithms (Adam and Momentum) in different datasets, namely MNIST, Fashion MNIST, and CIFAR-10. Specifically, we train a deep FNN, a convolution network, and the ResNet18[72] on MNIST, Fashion MNIST, and CIFAR-10, respectively. Each network training is performed with a supervised loss defined by a cross-entropy formula

$$H(x, y) = \sum_{k \in \mathcal{S}} -y_k \log \mathrm{NN}_\theta(x_k),$$

where x_k and y_k denote the input and the label of a sample (x_k, y_k). $\mathrm{NN}_\theta(x_k)$ is the output of NN for x_k. This cross-entropy loss is also used for assessing the performance of network models. More experimental details about networks and hyperparameters of optimization algorithms are presented in SI Section H.

Experiments on deep learning-based PDE solver

We mainly consider solving three benchmark PDEs by deep learning methods, i.e., the 1D Burgers' equation along with Dirichlet boundary conditions, the 2D incompressible NS equation on the square cavity (lid-driven cavity), and the 2D Darcy Flow equation on the unit box.

In the context of PINNs[68], the solutions of the Burgers' equation and the NS equation are approximated by FNNs, where the loss functions used for training can be referred to[73]. In particular, each network training is performed with a supervised loss taking the general form

$$\mathcal{L}_{\text{train}}(\theta) = \frac{1}{N_u} \sum_{i=1}^{N_u} \left\| \boldsymbol{u}_i - \mathrm{NN}_\theta(\boldsymbol{x}_i) \right\|^2 + \frac{1}{\lambda} \mathcal{R}\left[\mathrm{NN}_\theta(\boldsymbol{x})\right],$$

where $\boldsymbol{u}_i$ is the measurement data of the initial and boundary conditions, $\mathrm{NN}_\theta(\boldsymbol{x}_i)$ is the corresponding DNN-approximated solution, N_u is the total number of training data points, and $\mathcal{R}\left[\mathrm{NN}_\theta(\boldsymbol{x})\right]$ is designed to constrain the outputs of neural network to satisfy a PDE condition controlled by a parameter λ. To quantitatively assess the performance of network models, we adopt a relative L_2 norm as a test metric defined by

$$\mathcal{L}_{\text{test}}(\theta) = \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{\left\| \boldsymbol{u}_i - \mathrm{NN}_\theta(\boldsymbol{x}_i) \right\|^2}{\left\| \boldsymbol{u}_i \right\|^2},$$

where N_c is the total number of testing data points.

In addition, we utilize the FNO method[69] to solve the Burgers' equation and the Darcy Flow equation, where the network architectures used to approximate the PDE solution can be found in ref. 69. In these FNO experiments, we use a L_2 loss to train and test the network formulated by

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{u}_i - \mathrm{NN}_\theta(\boldsymbol{x}_i) \right\|^2.$$

In SI Section H, we provide more details about the networks and the hyperparameters of optimization algorithms. Here, we train the deep learning model with mini-batch algorithms, the mini-batch size of which is identical for all algorithms used in each example. We also note that with the given lr_{eq} and a_{eq}, the hyperparameters of PIDAO-family algorithms used in the above experiments (the classification and physical learning problems) follow one common criterion $h = lr_{eq}$, $a = \frac{1}{lr_{eq}}(\frac{1}{a_{eq}} - 1)$, $k_p = \frac{1}{a_{eq} lr_{eq}}$, $k_p > \frac{k_i}{a} + ak_d$ such that the equivalent momentum and lr in the PIDAO-family algorithm are equal to lr_{eq} and a_{eq}, respectively. To keep the fairness of comparison in experiments, the learning rate and the momentum coefficient of other algorithms (if any) are also equal to lr_{eq} and a_{eq}, respectively.

Subspace projection of loss landscape

In the MNIST-FNN case, we project the training and test loss landscapes (or surfaces) around the trained final parameter $\theta^\star$ (or θ_n) into low dimensional parameter subspaces using the visualization method[16]. More precisely, after obtaining the one-dimensional projections of the training and test loss landscape $f(\alpha; \theta^\star) = f(\theta^\star + \alpha\theta_{pd}), \alpha \in [-1, 1]$ with a specific projection direction θ_{pd}, we calculate the training and test accuracy along the projected training and test loss landscapes. Additionally, we denote θ_k as the model parameter vector at epoch k during the training. The training trajectory of each algorithm $\left[\theta_1, \theta_2, \cdots, \theta^\star\right]$ is also projected into a two-dimensional subspace $f(\alpha_1, \alpha_2; \theta^\star) = f(\theta^\star + \alpha_1\theta_{pd_1} + \alpha_2\theta_{pd_2})$ that is the projection of the train loss landscape. Here, after applying the principal component analysis to the matrix $M = \left[\theta_1 - \theta^\star, \theta_2 - \theta^\star, \cdots, \theta_{n-1} - \theta^\star\right]$ and obtaining the first second principal components, these two principal components become two projection directions θ_{pd_1} and θ_{pd_2}. It is worth mentioning that the first principal component can be also represented as the projection direction θ_{pd} used in the above one-dimensional projection.

Data availability

All benchmark datasets used in this paper are publicly available, including MNIST, FashionMNIST, CIFAR-10 that can be accessed in https://git-disl.github.io/GTDLBench/datasets/, and the PDE dataset for FNO that is available in https://drive.google.com/drive/folders/1UnbQh2WWc6knEHbLn-ZaXrKUZhp7pjt-.

Code availability

The source code[74] is available at https://github.com/NoulliCHEN/PIDAO.

References

1. O'Connell, M. et al. Neural-fly enables rapid learning for agile flight in strong winds. *Sci. Robot.* **7**, eabm6597 (2022).
2. Ichnowski, J., Avigal, Y., Satish, V. & Goldberg, K. Deep learning can accelerate grasp-optimized motion planning. *Sci. Robot.* **5**, eabd7710 (2020).
3. Loquercio, A. et al. Learning high-speed flight in the wild. *Sci. Robot.* **6**, eabg5810 (2021).
4. Kaufmann, E. et al. Champion-level drone racing using deep reinforcement learning. *Nature* **620**, 982–987 (2023).
5. Wurman, P. R. et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature* **602**, 223–228 (2022).
6. Iwami, R. et al. Controlling chaotic itinerancy in laser dynamics for reinforcement learning. *Sci. Adv.* **8**, eabn8325 (2022).
7. Böttcher, L., Antulov-Fantulin, N. & Asikis, T. AI Pontryagin or how artificial neural networks learn to control dynamical systems. *Nat. Commun.* **13**, 333 (2022).
8. Course, K. & Nair, P. B. State estimation of a physical system with unknown governing equations. *Nature* **622**, 261–267 (2023).
9. Han, J., Jentzen, A. & Weinan, E. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci. USA* **115**, 8505–8510 (2018).
10. Degrave, J. et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* **602**, 414–419 (2022).
11. Kates-Harbeck, J., Svyatkovskiy, A. & Tang, W. Predicting disruptive instabilities in controlled fusion plasmas through deep learning. *Nature* **568**, 526–531 (2019).

https://doi.org/10.1038/s41467-024-54451-3

12. Singhal, K. et al. Large language models encode clinical knowledge. *Nature* **620**, 172–180 (2023).
13. Lake, B. M. & Baroni, M. Human-like systematic generalization through a meta-learning neural network. *Nature* **623**, 115–121 (2023).
14. Bi, K. et al. Accurate medium-range global weather forecasting with 3D neural networks. *Nature* **619**, 533–538 (2023).
15. Liang, J., Xu, C. & Cai, S. Recurrent graph optimal transport for learning 3D flow motion in particle tracking. *Nat. Mach. Intell.* **5**, 505–517 (2023).
16. Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Visualizing the loss landscape of neural nets. In: *31st International Conference on Advances in Neural Information Processing Systems* (NeurIPS, Montreal, Canada, 2018).
17. Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. & Tang, P. T. P. On large-batch training for deep learning: generalization gap and sharp minima. In: *5th International Conference on Learning Representations* (ICLR, Palais des Congrès Neptune, Toulon, France, 2016).
18. Baldassi, C., Pittorino, F. & Zecchina, R. Shaping the learning landscape in neural networks around wide flat minima. *Proc. Natl. Acad. Sci. USA* **117**, 161–170 (2020).
19. Bottou, L., Curtis, F. E. & Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Rev.* **60**, 223–311 (2018).
20. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. In: *3rd International Conference on Learning Representations* (ICLR, San Diega, CA, USA, 2015).
21. Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* **4**, 1–17 (1964).
22. Jordan, M. I. Dynamical, symplectic and stochastic perspectives on gradient-based optimization, In *Proc. International Congress of Mathematicians: Rio de Janeiro 2018* 523–549 (World Scientific, 2018).
23. Tabuada, P. & Gharesifard, B. Universal approximation power of deep residual neural networks through the lens of control. *IEEE Trans. Autom. Control* **68**, 2715–2728 (2022).
24. Li, Q. & Hao, S. An optimal control approach to deep learning and applications to discrete-weight neural networks. In *Proc. International Conference on Machine Learning* 2985–2994 (PMLR, 2018).
25. Lessard, L., Recht, B. & Packard, A. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM J. Optim.* **26**, 57–95 (2016).
26. Hu, B. & Lessard, L. Control interpretations for first-order optimization methods. In *Proc. 2017 American Control Conference (ACC)* 3114–3119 (IEEE, 2017).
27. Fazlyab, M., Ribeiro, A., Morari, M. & Preciado, V. M. Analysis of optimization algorithms via integral quadratic constraints: non-strongly convex problems. *SIAM J. Optim.* **28**, 2654–2689 (2018).
28. Ruiz-Balet, D. & Zuazua, E. Neural ODE control for classification, approximation and transport. *SIAM Rev.* **65**, 735–773 (2023).
29. Shi, B., Du, S. S., Jordan, M. I. & Su, W. J., Understanding the acceleration phenomenon via high-resolution differential equations, *Math. Program.* **195**, 79–148 (2021).
30. Wilson, A. C., Recht, B. & Jordan, M. I. A Lyapunov analysis of accelerated methods in optimization. *J. Mach. Learn. Res.* **22**, 5040–5073 (2021).
31. Yang, L. et al. The physical systems behind optimization algorithms. In: *31st International Conference on Advances in Neural Information Processing Systems* (NeurIPS, Montreal, Canada, 2018).
32. Wibisono, A., Wilson, A. C. & Jordan, M. I. A variational perspective on accelerated methods in optimization. *Proc. Natl. Acad. Sci. USA* **113**, E7351–E7358 (2016).
33. Li, Q., Tai, C. & Weinan, E. Stochastic modified equations and adaptive stochastic gradient algorithms. In *Proc. International Conference on Machine Learning* 2101–2110 (PMLR, 2017).
34. Zhou, P., Feng, J., Ma, C., Xiong, C. & Hoi, S. C. H. Towards theoretically understanding why SGD generalizes better than Adam in deep learning. *Adv. Neural Inf. Process. Syst.* **33**, 21285–21296 (2020).
35. Da Silva, A. B. & Gazeau, M. A general system of differential equations to model first-order adaptive algorithms. *J. Mach. Learn. Res.* **21**, 5072–5113 (2020).
36. Barrett, D. & Dherin, B. Implicit gradient regularization. In: *9th International Conference on Learning Representations* (ICLR, Vienna, Austria, 2020).
37. Compagnoni, E. M. et al. An SDE for modeling SAM: theory and insights. In *Proc. International Conference on Machine Learning* 25209–25253 (PMLR, 2023).
38. Su, W., Boyd, S. & Candës, E. J. A differential equation for modeling Nesterov's accelerated gradient method: theory and insights. *J. Mach. Learn. Res.* **17**, 1–43 (2016).
39. Jacot, A., Gabriel, F. & Hongler, C. Neural tangent kernel: convergence and generalization in neural networks. In: *31st International Conference on Advances in Neural Information Processing Systems* (NeurIPS, Montreal, Canada, 2018).
40. Wang, S., Yu, X. & Perdikaris, P. When and why PINNs fail to train: a neural tangent kernel perspective. *J. Comput. Phys.* **449**, 110768 (2022).
41. Weinan, E. A proposal on machine learning via dynamical systems. *Commun. Math. Stat.* **1**, 1–11 (2017).
42. Xue, H., Araujo, A., Hu, B. & Chen, Y. Diffusion-based adversarial sample generation for improved stealthiness and controllability, In: *37th International Conference on Advances in Neural Information Processing Systems* (NeurIPS, New Orleans, Louisiana, USA, 2024).
43. Hauswirth, A., He, Z., Bolognani, S., Hug, G. & Dörfler, F. Optimization algorithms as robust feedback controllers. *Annu. Rev. Control* **57**, 100941 (2024).
44. Krstic, M., Kokotovic, P. V. & Kanellakopoulos, I. *Nonlinear and Adaptive Control Design* (John Wiley & Sons, Inc., 1995).
45. Sontag, E. D. *Mathematical Control Theory: Deterministic Finite Dimensional Systems* Vol. 6 (Springer Science & Business Media, 2013).
46. Ross, I. M. An optimal control theory for nonlinear optimization. *J. Comput. Appl. Math.* **354**, 39–51 (2019).
47. Li, Q., Chen, L., Tai, C. & Weinan, E. Maximum principle based algorithms for deep learning. *J. Mach. Learn. Res.* **18**, 1–29 (2018).
48. Xu, K., Li, C., Zhu, J. & Zhang, B. Understanding and stabilizing GANs' training dynamics using control theory. In *Proceedings of the 37th International Conference on Machine Learning* Vol. 119 of *Proceedings of Machine Learning Research* (eds Daumé, H., Singh, A.) 10566–10575 (PMLR, 2020).
49. Chen, Z., Li, Q. & Zhang, Z. Self-healing robust neural networks via closed-loop control. *J. Mach. Learn. Res.* **23**, 14329–14382 (2022).
50. Wang, S., Teng, Y. & Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.* **43**, A3055–A3081 (2021).
51. Kolarijani, A. S., Esfahani, P. M. & Keviczky, T. Continuous-time accelerated methods via a hybrid control lens. *IEEE Trans. Autom. Control* **65**, 3425–3440 (2019).
52. Vaquero, M. & Cortés, J. Convergence-rate-matching discretization of accelerated optimization flows through opportunistic state-triggered control. In: *32nd International Conference on Advances in Neural Information Processing Systems* (NeurIPS, Vancouver, Canada, 2019).
53. Ghosh, A., Lyu, H., Zhang, X. & Wang, R. Implicit regularization in heavy-ball momentum accelerated stochastic gradient descent. In: *11th International Conference on Learning Representations* (ICLR, Kigali, Rwanda, 2023).
54. Astrom, K. J. *PID Controllers: Theory, Design, and Tuning* (The Instrumentation, Systems and Automation Society, 1995).

55. Filo, M., Kumar, S. & Khammash, M. A hierarchy of biomolecular proportional-integral-derivative feedback controllers for robust perfect adaptation and dynamic performance. *Nat. Commun.* **13**, 2119 (2022).
56. Zhao, C. & Guo, L. Towards a theoretical foundation of PID control for uncertain nonlinear systems. *Automatica* **142**, 110360 (2022).
57. Ma, R., Zhang, B., Zhou, Y., Li, Z. & Lei, F. PID controller-guided attention neural network learning for fast and effective real photographs denoising. *IEEE Trans. Neural Netw. Learn. Syst.* **33**, 3010–3023 (2021).
58. Xu, J., Xiong, Z. & Bhattacharyya, S P. PIDNet: a real-time semantic segmentation network inspired by PID controllers. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* 19529–19539 (IEEE, 2023).
59. Wang, H. et al. PID controller-based stochastic optimization acceleration for deep neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **31**, 5079–5091 (2020).
60. Attouch, H., Chbani, Z., Peypouquet, J. & Redont, P. Fast convergence of inertial dynamics and algorithms with asymptotic vanishing viscosity. *Math. Program.* **168**, 123–175 (2018).
61. Attouch, H., Chbani, Z., Fadili, J. & Riahi, H. First-order optimization algorithms via inertial systems with Hessian driven damping. *Math. Program.* **193**, 113–155 (2022).
62. Karimi, H., Nutini, J. & Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Proc. European Conference on Machine Learning and Knowledge Discovery in Databases*, Vol. 9851, 795–811 (ECML, 2016).
63. Fazel, M., Ge, R., Kakade, S. & Mesbahi, M. Global convergence of policy gradient methods for the linear quadratic regulator. In *Proc. International Conference on Machine Learning* 1467–1476 (PMLR, 2018).
64. Liu, C., Zhu, L. & Belkin, M. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Appl. Comput. Harmon. Anal.* **59**, 85–116 (2022).
65. Li, Y., Ma, T., Zhang, H. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Proc. Conference On Learning Theory* 2–47 (PMLR, 2018).
66. Hairer, E., Hochbruck, M., Iserles, A. & Lubich, C. Geometric numerical integration. *Oberwolfach Rep.* **3**, 805–882 (2006).
67. Saab Jr, S., Phoha, S., Zhu, M. & Ray, A. An adaptive polyak heavy-ball method. *Mach. Learn.* **111**, 3245–3277 (2022).
68. Karniadakis, G. E. et al. Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
69. Li, Z. et al. Fourier neural operator for parametric partial differential equations. In: *9th International Conference on Learning Representations* (ICLR, Vienna, Austria, 2020).
70. Wu, L., Ma, C. & Weinan, E. How SGD selects the global minima in over-parameterized learning: a dynamical stability perspective. In: *31st International Conference on Advances in Neural Information Processing Systems* (NeurIPS, Montreal, Canada, 2018).
71. Nesterov, Y. *Introductory Lectures on Convex Optimization: a Basic Course* Vol. 87 (Springer Science & Business Media, 2003).
72. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 770–778 (IEEE, 2016).
73. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
74. Chen, S. et al. Accelerated optimization in deep learning with a proportional-integral-derivative controller. PIDAO, https://doi.org/10.5281/zenodo.13939583 (2024).
75. Loshchilov, I. & Hutter, F. Decoupled weight decay regularization. In 7th *Proc. International Conference on Learning Representations* (ICLR, New Orleans, Louisiana, United States, 2019).

Acknowledgements

C.X., S. Cai., and S. Chen would like to acknowledge the support by the National Natural Science Foundation of China (No. 61973270, No. 62301486, and No. 12171431). C.X. gratefully acknowledges the financial support provided by the Zhejiang Province High-Level Talent Special Support Plan for Scientific and Technological Innovation Leading Talents (No. 2023R5217). S. Cai acknowledges the support by Qizhen Scholar Foundation of Zhejiang University Education Foundation and the State Key Laboratory of Industrial Control Technology (Grant No. ICT2024A05). S.Chen acknowledges the scholarship of Doctoral Academic Star Program of Zhejiang University.

Author contributions

S. Chen performed theoretical analysis of the methods and wrote the original manuscript, assisted by J.L. S.Chen, P.W., C.X., and S. Cai implemented the algorithms, performed the experiments, and analyzed the results. S.Chen, J.L., P.W., C.X., S. Cai. and J.C. contributed to discussing the results, writing and reviewing the paper. C.X. and S. Cai supervised research.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41467-024-54451-3.

Correspondence and requests for materials should be addressed to Chao Xu or Shengze Cai.

Peer review information *Nature Communications* thanks the anonymous reviewers for their contribution to the peer review of this work. A peer review file is available.

Reprints and permissions information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

© The Author(s) 2024